sck cen

Registered Office
Herrmann-Debrouxlaan 40
1160 Brussel – Belgium

Foundation of Public Utility
VAT BE 406.568.867

Research Centres
Boeretang 200
2400 Mol – Belgium

Chemin du Cyclotron 6
1348 Ottignies-Louvain-la-Neuve – Belgium

Reference N°	Creation Date	
SCK CEN/49669088	2022-06-23	
Alternative Reference N°	Revision	Version
N/A	1.0	1
ISC	Revision Status	
Restricted	Approved	

ANICCA FUEL CYCLE IRRADIATION MODELS.pdf

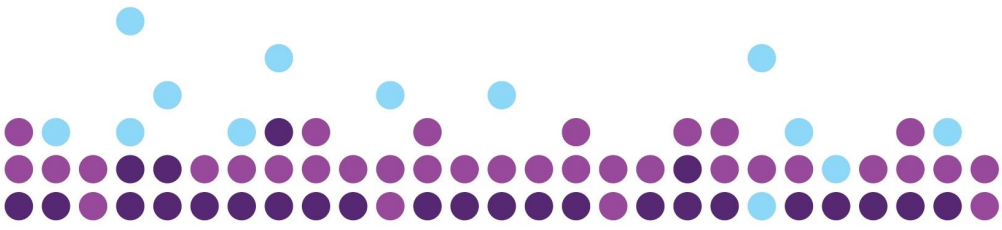
Final Thesis Document

Authors*
Victor Casas Molina

Approval information for current revision*		
Name	**Outcome**	**Date**
Gert Van den Eynde	Approved	2022-09-06

Change log*				
Revision	**Version**	**Status**	**Date**	**Description of change**
1.0	1	Approved	2022-06-23	

**This automatically generated cover page shows references and document information as were available in the Alexandria document management system on 2022-09-06. Please refer to Alexandria for current and complete metadata, or to the document contents and/or author for additional information.*



TRABAJO FIN DE MASTER

ANICCA FUEL CYCLE IRRADIATION MODELS:

A MACHINE LEARNER PREDICTOR AS A FUNCTION OF INITIAL FUEL COMPOSITION.

JUNIO 2022

Víctor José Casas Molina

TRABAJO FIN DE MASTER
PARA LA OBTENCIÓN DEL
TÍTULO DE **MASTER EN**
CIENCIA Y TECNOLOGÍA
NUCLEAR

DIRECTORES DEL TRABAJO FIN DE MASTER:

Augusto Hernández Solís (SCK-CEN)
Pablo Romojaro Otero (SCK-CEN)
Óscar Cabellos de Francisco (UPM)

INDUSTRIALES
ETSII | UPM

POLITÉCNICA

RESUMEN

Introducción

Los códigos de análisis de ciclo de combustible permiten evaluar la perspectiva general de una flota de reactores nucleares en un escenario determinado mediante el cómputo de los flujos e inventarios de combustible gastado, la potencia generada y los requerimientos en el suministro de combustible nuclear fresco. Dada su importancia en lo que respecta al panorama global de la energía nuclear resultan determinantes en la realización de valoraciones generales y estudios de idoneidad dentro de este campo.

ANICCA es el código de ciclo de combustible desarrollado por el Centro Belga para la Investigación Nuclear (**SCK-CEN**), el cual cumple los propósitos anteriormente mencionados, mediante la simulación versátil de escenarios en los que se definen diferentes conexiones entre las múltiples instalaciones modelables en éstos, como por ejemplo, reactores de potencia, repositorios de almacenamiento final de residuos, minas de mena de uranio, plantas de reprocesamiento, etc.

Este código ya ha sido empleado para la simulación de benchmarks donde han participado otros códigos como COSI6, FAMILY-21 o TR_EVOL. Se ha empleado también en el cálculo y simulación de escenarios avanzados como es el caso del escenario de cese de explotación nuclear belga, en el que mediante el modelado de reactores avanzados como el ADS (*Accelerator-driven System)*, se pretendía estudiar la transmutación de actínidos menores provenientes de la explotación de los reactores convencionales.

El objetivo de este trabajo, desarrollado como consecuencia de una estancia en el SCK-CEN, es el de implementar mejoras en dicho código mediante técnicas de inteligencia artificial. Concretamente es objeto de modificaciones el módulo de irradiación de ANICCA, a través del cual se obtiene, entre otras cosas, el inventario isotópico final tras la irradiación de combustible fresco.

El módulo original de irradiación de ANICCA está basado en la aplicación del método de la aproximación racional de Chebyshev (CRAM), el cual se emplea para resolver las ecuaciones de Bateman con el objetivo de realizar el proceso de quemado.

Los datos requeridos por el CRAM se proporcionan al código a través de unas **librerías generadas** *a priori* por un código basado en Monte Carlo como SERPENT2 o ALEPH2. Dichas librerías contienen cada una las secciones eficaces requeridas para realizar los cálculos para un quemado único, estas secciones son colapsadas en un grupo energético y son además promediadas en el tiempo, por lo que se realizan a través de este método dos fuertes simplificaciones para la obtención del inventario final. Cabe también mencionar, la patente rigidez del método, pues en el caso de existir en el escenario más de un reactor con diferentes quemados objetivo (o diferentes tipos de combustibles), se requerirá generar nuevas librerías con dichos quemados objetivo para un tipo de combustible específico.

Revisando el estado del arte, se han encontrado soluciones muy alineadas con los objetivos del presente trabajo relacionadas con el **Deep Learning**, como es el caso de una publicación en la que se describe el empleo de redes neuronales para una aplicación similar sobre el código de ciclo francés CYCLUS.

Estudiando los diferentes algoritmos de aprendizaje automático, se ha decidido implementar un modelo basado en **redes neuronales**, en el cual el output sea la densidad de ciertos isótopos importantes para el análisis de ciclo de combustible y la caracterización del combustible nuclear gastado. Para ello, se emplean como inputs variables que se pueden obtener con relativa facilidad del código fuente de ANICCA, a saber: el enriquecimiento inicial (o contenido inicial de plutonio si se trata de un combustible tipo MOX) y el quemado alcanzado en el combustible a la descarga.

El motivo de emplear redes neuronales estriba en la necesidad de efectuar **un problema de regresión múltiple no lineal** con el objetivo de definir las relaciones entre las variables independientes (quemado y enriquecimiento) y las variables dependientes (densidad final de los isótopos estudiados en el inventario final). Sumando a esto la posibilidad de generar datos suficientes como para entrenar el modelo, el problema se centraba básicamente en un caso de aprendizaje supervisado, y como matiz adicional, se destaca el hecho de que el número de dependencias a encontrar es sustancial cuando se desean predecir un número importante de isótopos (en este caso hasta un total de 73). Por todo ello, la elección final fue la de emplear dichas redes neuronales.

Con objetivo de acotar el alcance del trabajo, se concluye que solo se realizará la implementación de los modelos de redes neuronales para **reactores tipo PWR**, y al mismo tiempo solo se realizará para **combustibles UOX y MOX**, dejando fuera del alcance otros combustibles que sí son procesables por ANICCA como combustibles tipo MOXEU. El motivo de esta simplificación es que el trabajo pretende realizar la implementación a modo de demostración tecnológica, y las asunciones y conclusiones extraídas son fácilmente extrapolables a otras tecnologías y/o combustibles.

Redes neuronales

Las redes neuronales artificiales, o por sus siglas en inglés ANN, son un conjunto de algoritmos que emulan el comportamiento del cerebro humano mediante un modelado matemático simplificado basado en las neuronas cerebrales.

Estos algoritmos se pueden entender como una caja negra en la que se ingresa un input n-dimensional y se devuelve un output m-dimensional. Aunque tienen múltiples aplicaciones dependiendo del tipo de problema y de la topología de esta (reconocimiento de imágenes, clasificaciones, predicciones, etc.), se presentan únicamente aquella centrada en el problema objeto de este trabajo: redes neuronales directas (*Forward Neural Networks*) para problemas de regresión.

Estas redes neuronales están **formadas por capas de perceptrones** o neuronas, las cuales se interconectan entre sí de manera que cada neurona de una capa **i** esté conectada con las neuronas pertenecientes a la capa **i-1** e **i+1**. Entre estas capas existe una distinción que las clasifica en tres tipos atendiendo a su posición en la arquitectura de la red: la capa de entrada (*input layer*), las capas intermedias u ocultas (*hidden layers*) y la capa final o la capa de salida (*output layer*). Las capas de entrada y salida son los elementos de ingreso y terminal de salida de los datos respectivamente, su función se limita pues, a transmitir la información a las capas centrales.

Las conexiones neuronales que tejen la red son, desde un punto de vista matemático, multiplicaciones por unos parámetros únicos para cada conexión, de esta manera, se pueden manipular los inputs mediante la ponderación realizada por las diferentes conexiones, cuando las neuronas de cada capa reciben el valor ya ponderado por las conexiones antecedentes, estas suman los inputs recibidos y generan un output único y exclusivo de cada neurona que se envía a todas las neuronas de la siguiente capa (se remarca de nuevo que solo se están tratando las redes neuronales directas, lo afirmado anteriormente es cierto para esta clase de redes, pero no para otras como las redes neuronales recurrentes).

Las neuronas por tanto básicamente cumplen la función de sumar sus inputs y reenviarlos a la siguiente capa mediante un único output, no obstante, antes de ser expedidos los valores por la salida de la neurona, son convolucionados en una función de activación.

Hasta ahora, si se escribe en forma analítica el resultado equivalente de los procesos descritos se obtendrá siempre una función linear, pues solo existen unos valores que aparecen por la capa de ingreso (escalares) que son multiplicados por un parámetro de ponderación dictado por cada conexión entre neuronas (también un escalar), por tanto, y como la composición de una función lineal es siempre una función lineal, se requiere la existencia de una función de activación a la salida de cada neurona para inducir cierta no linealidad tratando así problemas complejos distintos a la regresión lineal. En este caso, se han empleado como funciones de activación funciones tipo ReLU, o *Rectified Linear Unit,* la cual da valores nulos para un valor negativo y valores de acuerdo a una recta $y = x$ para valores mayores que cero. Estas funciones reciben el nombre del concepto de activación neuronal, pues es condición necesaria superar el umbral impuesto por esta para que esa neurona en concreto se “active”.

Explicada la arquitectura general de una red neuronal directa, se pasan a explicar los **mecanismos de aprendizaje**, los cuales se basan en dos algoritmos: *Forward Propagation* y *Back Propagation*. El primero explica cómo la información viaja desde la entrada hasta la salida, transformando esta primera mediante la realización de las pertinentes operaciones básicas. Una vez obtenido un primer resultado tentativo en la salida, se computa el error entre el punto predicho y el real, este error se utiliza en algoritmos que lo disminuyen, como el de descenso del gradiente estocástico para así actualizar los diferentes pesos de las distintas conexiones de la arquitectura, esta actualización de pesos se propaga hacia atrás mediante el segundo algoritmo (*Back Propagation)*.

Del párrafo anterior se deduce que se requiere de al menos dos cosas: una cantidad suficiente de datos para entrenar el modelo de manera que se alcance un error lo suficientemente reducido para hacerlo válido, y que dentro de estos datos existan también los resultados esperables para cada input proporcionado. Por ello la información que se provea al modelo debe estar etiquetada y separada en características, inputs o *features*, y en salidas u *outputs.*

El entrenamiento de la red puede suponer un problema si la máquina “aprende” del ruido de los datos de manera accidental, causando una condición conocida como *overfitting*, en la que el modelo parece predecir bien durante el entrenamiento, pero luego tiene poca eficacia con datos no observados durante el entrenamiento, por ello, es importante partir los datos en un set de entrenamiento (*train set)* y un set de comprobación (*test set)*, con el objetivo de evaluar lo bien que está prediciendo el modelo en cuestión.

Investigación preliminar

En primer lugar, se decidió usar el entorno de **Keras** y por tanto el lenguaje de programación **Python** para la realización de las tareas que componen el presente trabajo. Esta conclusión emana tanto del hecho de que Keras está basado en Python, así como de que ANICCA también está programado en dicho lenguaje.

Para realizar una correcta evaluación de las necesidades de los modelos se ha realizado una investigación preliminar a partir de un set de datos existente facilitado por la Universidad de Uppsala. Este consistía en los datos de quemado generados con **SERPENT2** para un total de 279 isótopos, en los que se disponía como inputs asimilables para la red neuronal el enriquecimiento inicial, el tipo de combustible (UOX o MOX), el quemado de descarga y el tiempo de enfriamiento.

Se realizaron sobre este set pruebas preliminares con redes neuronales en las que se determinó cómo proceder en la generación definitiva de los modelos. Las principales conclusiones alcanzadas fueron:

1. Se empleará de la misma manera el código de transporte de Monte Carlo SERPENT2 para la generación de los set de datos para UOX y MOX.
2. Cada tipo de combustible estudiado tendrá su propio modelo, es decir, existirán dos redes neuronales separadas e independientes.
3. El tiempo de enfriamiento se descarta como input en el modelo debido a que actualmente ANICCA no maneja ninguna variable representativa de dicho parámetro que se pueda transferir de manera simple y ubicua hacia la entrada de los modelos.
4. Como consecuencia del punto anterior, se escoge realizar los cálculos de quemado sin tiempo de enfriamiento, puesto que si no se discrimina esta variable, aparecen distribuciones multimodales intratables por una arquitectura de red de dimensión 2 a dimensión 73, estas distribuciones bimodales aparecen especialmente en isótopos que se queman durante el arranque del reactor o que decaen rápidamente como es el caso del Xe 135.
5. Actualización de las librerías de datos nucleares empleada, se evita el uso de la librería JEFF-3.3 para las secciones eficaces para neutrones debido a comportamientos anómalos en la constante de multiplicación infinita en problemas de quemado cuando se compara con otras librerías, especialmente este efecto se aprecia para el Pu-239, por lo que se decide ejecutar los cálculos empleando ENDF/B-VIII.0.
6. Actualización de la versión de SERPENT2 empleada (2.1.28 a 2.1.32).

Elección de los isótopos

Los **73 isótopos objeto de estudio** han sido extraídos de referencias bibliográficas dedicadas a la caracterización de combustible gastado y al análisis de ciclo de combustible, concretamente de:

- 2018 JRC technical report: *Observables of interest for the characterization of Spent Nuclear Fuel.*
- 2009 Idaho National Laboratory: *Selection of Isotopes and Elements for Fuel Cycle Analysis.*

Entre los 73 isótopos elegidos se encuentran diferentes contribuyentes a la generación de calor (Pu-238, Pu-239), la emisión de rayos gamma (Cs-137), la emisión de neutrones (Am-241, Cm-244), indicadores de quemado (Nd-148) así como otros elementos interesantes como productos gaseosos, residuos de larga vida media, indicadores de burnup credit y aquellos elementos pertenecientes a las cadenas del uranio y el plutonio.

Generación de la base de datos

Previamente a la generación de los sets de datos para entrenar y testear las redes neuronales, se realiza una evaluación de las incertidumbres obtenidas entre el nuevo modelo generado en SERPENT2 y el modelo de referencia del repositorio de la Universidad de Uppsala. El objetivo es comprobar la validez de la geometría empleada en SERPENT2, así como obtener confirmación de que el modelo es suficientemente bueno en base a la reproducibilidad de los datos como para usarlo de manera definitiva.

Una vez realizada esta preparación previa, el input de SERPENT2 que generaba los datos mediante la simulación sobre una pin-cell, se ejecuta en serie mediante un script realizado en Python que permitió correr los inputs necesarios para cubrir los valores escogidos de enriquecimiento para ambos tipos de combustible estudiados. La ejecución de estos inputs tardó, de manera aproximada, dos semanas en finalizar. En consecuencia, se genera una base de datos con las siguientes características:

DATASET	***UOX***	***MOX***
ENRIQUECIMIENTO	1.5% a 6.0% (en pasos de 0.025%)	4.0% a 10.0% (en pasos de 0.025%)
QUEMADO	5 a 70 MWd/kgHM (pasos de 0.2 MWd/kgHM)	
CICLO DE IRRADIACIÓN	30 días a potencia cero cada 10 MWd/kgHM	
TRAIN/TEST SPLIT	70/30	
DATOS EN EL DATASET	63 712	84 832

Los valores que definen el set de datos pretenden cubrir los **valores típicos de enriquecimiento** o contenido de plutonio y quemado de los combustibles estudiados. Además se simula un parón en el ciclo de irradiación cada 10 MWd/kgHM para tener en cuenta los ciclos de recarga del combustible en un reactor PWR convencional. Los pasos de quemado se inician desde cero por motivos de precisión, pero debido a que no se requiere predecir combustible fresco, se descartan valores bajos de quemado en la confección del set de datos. El split, es la proporción en la que se dividen de manera aleatoria las 63 712 filas generadas para UOX y las 84 832 filas generadas para MOX, de tal manera que 70% de ellas servirán para entrenar el modelo y el 30 % restante servirán para realizar pruebas de ajuste sobre dicho modelo.

Finalmente y como conclusión de esta parte del trabajo, se obtienen cuatro archivos .csv en los que se almacenan los distintos valores generados para UOX y MOX respectivamente, existiendo para combustible un archivo con los datos de entrenamiento y otro con los de comprobación. Existiendo 73 isótopos y dos inputs, estos archivos disponen por lo tanto de 75 columnas cada uno.

Tratamiento previo de los datos

Debido al funcionamiento de las funciones de activación, las cuales están acotadas en este caso de 0 a 1, es necesario normalizar los datos para no saturar las funciones de activación. Además, debido al quemado intermitente que experimenta el Sm-149 durante las paradas simuladas en SERPENT2, se tienen numerosos puntos en los valores de quemado múltiplos de 10 MWd/kgHM, por lo que se ha optado también por curar dichos puntos y eliminarlos del dataset debido a que agregaban ruido a las predicciones tal y como se pudo comprobar en un ensayo preliminar.

Generación de la arquitectura de las redes

Curados y preparados los datos, se requiere **encontrar una arquitectura** para cada red neuronal que optimice el proceso de predicción. Para ello se recurre al ajuste de **hiperparámetros**. Dado lo expuesto anteriormente acerca del funcionamiento de las redes neuronales, se concluye que mediante la minimización del error los parámetros de cada conexión se actualizan hasta alcanzar valores tal que la predicción de la red se acerque cada vez más a los valores reales. No obstante, existen parámetros que no se actualizan durante el proceso de entrenamiento debido principalmente a que son estos mismos los que definen el problema, siendo por así decirlo las condiciones iniciales de este. Estos parámetros reciben el nombre de hiperparámetros debido a su naturaleza, y entre ellos destacan algunos como el número de neuronas por capa, el número de capas intermedias u ocultas o la tasa de aprendizaje.

La tasa de aprendizaje mide la rapidez con la que se va a ejecutar el algoritmo optimizador que persigue la minimización del error. Una tasa de aprendizaje muy alta permitirá al optimizador "saltar" entre mínimos locales pudiendo no encontrar los valores mínimos de error, o incluso llegar a mostrar un comportamiento divergente, lo que se conoce como *overshooting*. En cambio, una tasa de aprendizaje muy baja se manifiesta con un aumento de la demanda computacional. Existe pues un compromiso entre los recursos computacionales disponibles y la precisión requerida en la red. Para conseguir ajustar estos parámetros se recurre a optimizadores que, de manera iterativa, prueban diferentes valores para estos usando como métrica el error alcanzado para un set de datos dado, obteniendo las diferentes combinaciones de hiperparámetros que alcanzan un error más bajo para ese set de datos concreto. No obstante, si el espacio de búsqueda de las variables es muy grande, se tendrán de nuevo problemas relacionados con el elevado tiempo de cálculo, por ello existen diferentes optimizadores que se aplican a estas búsquedas denominados *tuners*. De entre todos los optimizadores se han probado los siguientes:

1. **Grid Searc**h: de búsqueda organizada sobre un espacio de búsqueda definido.
2. **Bayesian Search:** Se escoge el valor a testear en función del desempeño del anteriormente escogido, ideado para reducir el tiempo de búsqueda.
3. **Hyperband:** optimizador de la suite de Keras capaz de ajustar las épocas (veces que se procesan el total de los datos de entrenamiento) en las que entrena el modelo para una configuración dada en base al hecho de que la mayor disminución del error se logra en las primeras etapas del entrenamiento.

Finalmente, se ha escogido el **optimizador *Hyperband*** por ser el que menos error ha obtenido en las métricas durante el proceso de comparación entre optimizadores para estos sets de datos dado. Como resultado de la ejecución de *Hyperband*, se obtienen los siguientes hiperparámetros para la red neuronal de UOX:

Número de capas ocultas	***6***
Número de neuronas en la capa 0	44
Número de neuronas en la capa 1	140
Número de neuronas en la capa 2	236
Número de neuronas en la capa 3	236
Número de neuronas en la capa 4	80
Número de neuronas en la capa 5	176
Tasa de aprendizaje	**0.001**
Puntuación (Error Cuadrático Medio)	**1.825e-05**

Procediendo de la misma manera se obtienen los siguientes hiperparámetros para la red neuronal de MOX:

Número de capas ocultas	***5***
Número de neuronas en la capa 0	56
Número de neuronas en la capa 1	32
Número de neuronas en la capa 2	104
Número de neuronas en la capa 3	284
Número de neuronas en la capa 4	32
Tasa de aprendizaje	**0.0001**
Puntuación (Error Cuadrático Medio)	**2.027e-05**

Aparte de estos valores anteriormente recogidos en las tablas, se han escogido y ajustado manualmente el inicializador de parámetros y el optimizador de la red (que no está relacionado con el *tuner* de los hiperparámetros). Cuando el problema se inicializa, se requiere producir valores iniciales para todos los parámetros de las conexiones neuronales, para ello se usan diferentes distribuciones, en este caso se ha empleado una distribución normal modificada denominada *HeNormal,* la cual según la literatura revisada, obtiene buenos resultados con funciones de activación tipo ReLU.

Como optimizador del error se ha escogido el denominado como Adam, el cual es uno de los más avanzados y con mejores resultados de acuerdo con el estado del arte de la cuestión. Este optimizador es capaz de adaptar la tasa de aprendizaje conforme avanzan las épocas de entrenamiento para ajustarla conforme al progreso realizado en el entrenamiento. Adam se puede entender como una variación del clásico método del descenso estocástico del gradiente.

Tras el entrenamiento se comprueba que no existen problemas de *overfitting*, es decir, se comprueba que el modelo no ha aprendido del ruido presente en el set de datos. Debido a que los datos se han generado de manera artificial, sistemática y a partir de un código, no existen errores ni ruidos provenientes de mala catalogación de las etiquetas u otros errores comunes en la cadena de adquisición de datos, por lo que no se esperaban inicialmente problemas de *overfitting*.

Una vez entrenados y probados los modelos contra el output de SERPENT2, se guardan estos en dos modelos en formato hdf5 con el objetivo de integrarlos en el código ANICCA, además se requiere guardar la información de la normalización realizada para poder tratar posteriormente la información predicha dentro de ANICCA. Para ello, se exportan los scalers de entrada y salida en formatos tipo pickle. En total se generan 6 archivos, 3 para la red de UOX y otros 3 para la red de MOX, uno conteniendo el modelo de la red, y los otros dos la información de normalización de los inputs y de los outputs.

Integración en ANICCA

ANICCA está programado originalmente aprovechando la flexibilidad que permite un lenguaje orientado a objetos como Python, luego una vez que se tiene una idea sobre el diagrama de bloques y de flujo del código, es relativamente sencillo realizar modificaciones.

En este caso, **se ha agregado una nueva lista de isótopos** aparte de todas las listas de observables que existían originalmente. La nueva lista está compuesta por los 73 isótopos objeto de estudio; de esta manera, se puede conseguir que tras la ejecución del programa solo se hayan procesado los isótopos pertinentes.

Al mismo tiempo **se han generado dos variables** que redirigen el enriquecimiento inicial del combustible y el quemado de descarga hasta los modelos de las redes neuronales, los cuales se cargan desde el módulo de irradiación de ANICCA. Para el quemado, se ha tenido que realizar la suma de los pasos de cada librería existente en ANICCA, no obstante, si la implantación fuera definitiva, se requeriría definir en el input una variable a través de la cual se insertase el quemado de descarga, esta es pues una solución de carácter provisional. Como los modelos de las redes se han importado desde Keras, se requiere que se cargue como paso previo a la predicción, los módulos de Keras y de Tensorflow que se requieran para el funcionamiento de sendos modelos.

Finalmente, una sencilla estructura de control de flujo arbitra entre la activación de un modelo u el otro, con el objetivo de discriminar entre el tipo de combustible que se esté prediciendo en el caso de estar simulando un escenario de ciclo cerrado en el que ambos tipos de combustible intervengan.

Diseño del Benchmark

Para comprobar que la **integración con ANICCA** ha sido exitosa, se diseñan dos escenarios diferentes: uno de ciclo abierto y otro de ciclo cerrado. Así se puede verificar la red de UOX por separado, la red de UOX y MOX funcionando de manera conjunta, y descartar la existencia de algún problema que surja de la interacción entre ambas.

El escenario de ciclo abierto de combustible está compuesto por dos reactores PWR, de los cuales se estudian los flujos de combustible y residuos durante un lapso de 400 años. El de ciclo cerrado es ciertamente más complejo y se compone de siete reactores, dos de los cuales tienen capacidad de emplear MOX proveniente del reprocesado del combustible gastado.

Resultados

Tras la comparación de los resultados obtenidos contra SERPENT2 y contra ANICCA, se encuentra que se obtiene una capacidad de predicción razonablemente buena por parte de los dos modelos.

En lo que respecta a los ensayos realizados sobre el **escenario de ciclo abierto** en el que se estudian **el plutonio y uranio totales, los actínidos minoritarios, los elementos transuránicos y los productos de fisión en el inventario final**, se encuentra que se obtienen para todos los grupos estudiados desviaciones menores al 10%, con la excepción de los actínidos menores, para los que se obtienen desviaciones de hasta el 15%. De la misma manera, se observa una tendencia idéntica en el caso del **ciclo cerrado**, donde se vuelven a obtener valores de un 10% de desviación máxima para el plutonio, el uranio, los productos de fisión y los elementos transuránicos, y un 20% de desviación máxima para los actínidos minoritarios. Dado que la desviación se da en forma de error relativo, es necesario remarcar que el grupo de los actínidos minoritarios tienen isótopos que se dan en pocas cantidades en el inventario final para quemados convencionales, por lo que es de esperar que el error relativo sea sensiblemente más alto que para otros isótopos con mayor abundancia relativa.

Si se comparan los resultados predichos contra los reales para todos los puntos del conjunto de datos para los dos combustibles, se encuentra que para **el 80% de los isótopos** siempre se cumple que estos tienen menos de un 10% de error para quemados mayores que 25 MWd/kgHM. Al mismo tiempo, se comprueba que se obtienen los valores máximos de error para isótopos muy pesados (A>246), los cuales en particular son los causantes del aumento en la desviación de los actínidos minoritarios en los benchmarks realizados.

El hecho de tener valores de error más elevados para esos isótopos más pesados se debe parcialmente a que la abundancia relativa de estos para todos los puntos estudiados es casi nula, agrupándose la distribución de estos valores en torno al cero dando posiblemente problemas relacionados con la saturación de las redes neuronales.

En general, los **valores predichos con menor error** son aquellos que no tienen valores de quemado o enriquecimiento en las zonas limítrofes del espacio que compone el set de datos, es decir, las desviaciones más grandes se obtienen para valores de quemado y enriquecimiento próximos al máximo y mínimo del set de datos. Esto a su vez parece no ser un problema en la aplicación práctica de las redes neuronales, ya que la mayoría de los reactores tipo PWR convencionales funcionan dentro de estos valores y muy alejados de los valores extremos del set de datos aquí utilizado.

También se han evaluado el **desempeño del módulo** de irradiación tradicional y el de la actualización con los modelos nuevos en términos de aprovechamiento de recursos computacionales. La prueba más significativa realizada es quizás la de tiempo de cómputo total requerido por las dos versiones, obteniéndose que para el ciclo cerrado de combustible **se finaliza el cálculo 100 segundos antes**, lo que equivale a una reducción de casi el 50% del tiempo de cómputo para escenarios complejos. No obstante, para el ciclo abierto, se ha obtenido que el modelo basado en redes neuronales se demora 10 segundos más. Aun así este empeoramiento parece ser fácilmente enmendable, pues se debe posiblemente a cómo se importan los módulos requeridos por los modelos en el programa. Se considera fuera del alcance de este proyecto la optimización del código generado.

Se concluye finalmente que los esfuerzos realizados han contribuido a actualizar el módulo de irradiación, así como a demostrar la viabilidad de esta tecnología para aplicaciones relacionadas con el estudio del ciclo de combustible nuclear.

Como contrapartida, se observa que las redes neuronales pueden ser realmente herramientas muy rígidas, por ejemplo, para **el caso del MOX**, se ha requerido fijar el vector de plutonio contenido por el combustible fresco para generar el set de datos, por lo que cambiar dichas proporciones en el caso de que se requiera simular otro tipo de MOX, requeriría generar de nuevo un set de datos y regenerar por tanto la arquitectura de la red de MOX.

Para mitigar estos inconvenientes, se han propuesto **trabajos futuros** como una mejora en el despliegue de las redes neuronales en ANICCA con el objetivo de acelerar el proceso de irradiación, así como la generación de un modelo que se sirva de un set de datos más extenso en el que existan como características o inputs de la red neuronal la concentración de los diferentes isótopos que se puedan encontrar en el combustible fresco así como el quemado de descarga, el problema en este caso es el extenso periodo de cálculo que tomaría a SERPENT2 realizar todos los cálculos de quemado requeridos. Por ello en este caso, se ha tenido que descartar la idea en una primera aproximación debido a la falta de capacidad computacional que permitiera generar dicho set de datos en un tiempo razonable.

Finalmente, se hacen constar las modificaciones realizadas sobre los módulos de ANICCA y la distribución de error obtenida del benchmark contra SERPENT2 de los isótopos estudiados, recogidas en los apéndices A y B de este documento, respectivamente.

ABSTRACT

Machine Learning and Deep Learning techniques are gaining popularity due to the numerous applications they serve, being one of this applications the use of neural networks for predicting values from input data. This implies the possibility of replacing relatively complex or computationally expensive mathematical models with these neural networks.

Therefore, the present work, developed at the Belgian Nuclear Research Centre (SCK-CEN), has been focused on the use of these neural networks for the substitution of more complex processes. In this case, the substitution of the Chebyshev Rational Approximation Method (CRAM) was achieved. CRAM is used in the in-house Python-coded ANICCA nuclear fuel cycle simulator to solve the Bateman depletion equations to calculate the final isotopic inventory after an irradiation process.

The main purpose of this work was to replace the ANICCA irradiation module that hosted the CRAM with a model based on these neural networks that would have as input the initial fuel enrichment (or plutonium content if MOX) and discharge burnup, since such information is easily retrievable from within the current ANICCA code.

For this purpose, two neural networks were generated, one for cycles involving conventional Uranium fuel, and one for those using MOX fuel. These networks were trained from two datasets generated from depletion calculations performed in SERPENT2. These datasets encompassed many of the normal enrichment values for UOX and plutonium content for MOX, as well as numerous burnup steps covering all conventional discharge burnup values.

After obtaining the data and optimizing the hyperparameters and then training the neural networks in the Keras API for Deep Learning, the neural networks were able to predict in the final inventory the mass density of 73 isotopes relevant for spent nuclear fuel characterization and fuel cycle analysis. The observed deviations between SERPENT2 and the output of the networks were less than 10% for 80% of the tested values. However, some greater deviations were found for heavy nuclides such as Californium and Curium.

The neural network models were then saved and deployed in ANICCA's irradiation module to test the performance of these models by means of a benchmark in scenario studies. These benchmarks resulted in a successful integration of the modifications, with deviations of less than 10% for the final inventories of Uranium, Plutonium, Fission Products and Transuranium elements. However, a slightly higher deviation was observed for the minor actinides. At the same time, computation time was reduced by a solid 50% for complex scenarios.

However, the rigidity of this approach was highlighted since the MOX model can only infer the inventory for fuels with a single plutonium vector and can only predict as many isotopes as the network has been trained for. To address these limitations, new neural networks can be trained, improving the accuracy of predictions by adding more inputs, such as the cooling time to deal with short-lived isotopes, and the weight percentage of different isotopes in the fresh MOX fuel to predict several MOX compositions. Finally, computational time could also be enhanced by applying different and more advanced and specific deployment and coding techniques for the neural networks design.

UNESCO CODES: 1203.04, 2207.13, 2207.15

KEYWORDS: Machine Learning, Deep Learning, Neural Network, Fuel cycle, Nuclear, Isotopic Inventory, MOX, UOX, Spent Nuclear Fuel, ANICCA, SCK-CEN,

ACKNOWLEDGEMENTS

The author would like to personally thank his mentors at SCK-CEN Pablo Romojaro and Augusto Hernández, as well as Professor Óscar Cabellos of the Universidad Politécnica de Madrid for his guidance during the preparation of this thesis. In addition, he would like to thank Zsolt Elter and Riccardo Rossa for their insightful advice regarding the machine learning aspects.

TABLE OF CONTENTS

1. INTRODUCTION

Many countries are currently revisiting their energy policies given the rising prices, uncertainties and volatility of some of the available options for energy production [1],[2]; in Europe, the recent surge in gas prices has induced a raise in electricity prices [3],[4]. Nevertheless, it is not a condition only to be expected in gas supply; in fact, every fossil fuel is particularly sensitive to these kinds of problems given that there exists a notable geographical asymmetry between producers and consumers in relation to energy resources [5]–[7].

Thus, energy dependency reduction is posed as a mitigation measure against supply prices and availability issues. Nuclear energy is one of the most important shares of the energy mix pertaining to this strategy. Nonetheless, it is still a contentious technology among the population. However, it has been regaining its reputation due to the new technology developments, and specially, due to the arising energy crisis, at least concerning the European spheres [1].

Recently, The EU has endorsed nuclear as a green transition energy, but there are still many flaws in the whole chain of its technology to consider, specially taking into account that nuclear energy is always a politically-hued matter [8].

Among all the methodology the countries have in their repertoire, it shall be specially mentioned the potential that fuel cycle simulations could bring.

The nuclear power supply chain management is an arduous task to be performed and simulations involved are ubiquitous in order to assist in the management plan for supply, technology deployment, and final waste disposal or reprocessing. Different scenarios are estimated in order to enhance the reliance on the decisions concerning the energy strategy in each country.

Fuel cycle codes are capable of assessing the general perspective of a whole nuclear fleet; these tools can model various nuclear fuel cycle possibilities in a different kind of nuclear reactors. The feedback is then employed to accurately identify the supply needs and the required infrastructure for a given nuclear fleet, acting as an accurate estimator, or it could be employed to compare different technologies in terms on costs and time [9]–[11].

Then, special attention should be paid to these simulator capabilities regarding the back end of the fuel cycle. The disposal of the spent nuclear fuel is usually a controversial aspect of nuclear energy among both public and academia, and it is seen as one of the main drawbacks of this technology by the public.

Consequently, in terms of the aforementioned aspects, it could be stated that these fuel cycle codes are a useful tool to redesign a national energy strategy that involves nuclear power as well as a potential political and legislative decision tool.

In this thesis the scope is focused on the Belgian fuel cycle code ANICCA, which is the in-house code of the Belgian Nuclear Research Center SCK-CEN. Specifically an approach to address an alternative to its current irradiation model is presented. This is the block with the most influence on the final isotopic composition of the SNF (*Spent Nuclear Fuel)*, which is a significant aspect regarding the nuclear fuel back end.

1.1. The ANICCA fuel cycle scenario code

The **A**dvanced **N**uclear **I**nventory **C**ycle **C**ode: **ANICCA** is the fuel cycle scenario code that is being developed at SCK-CEN. It was conceived to be as an easy-to-use code, with a hands-on approach: easy to maintain and easy to extend. ANICCA is scripted on Python, an election based on its readability and its increasing and widespread use [12], [13].

It has been conceived as a modular code to achieve flexibility while remaining as generic as possible. Given this modular building, the core of the code capabilities can be enhanced by using different packages which are loaded automatically if they are invoked; this means, they are needed because of the task required by the user.

One of its main features is the capability of running depletion calculations without needing external codes or processes. Consequently, decay and burning processes of nuclear materials are executed within the code.

The aforementioned nuclear materials could be a variety of different fuel types such as MOX and LEU, which can be assigned to different fuel cycles (open, closed, etc.) and to different burning strategies (100% UOX core, MOX in core, MOXEUS[1], etc.) [13].This is achieved by using an approach assumed by the code called Scenario, in which the different facilities (also a tunable object from the code called Component) are linked each other via connections embracing that way the whole fuel cycle:

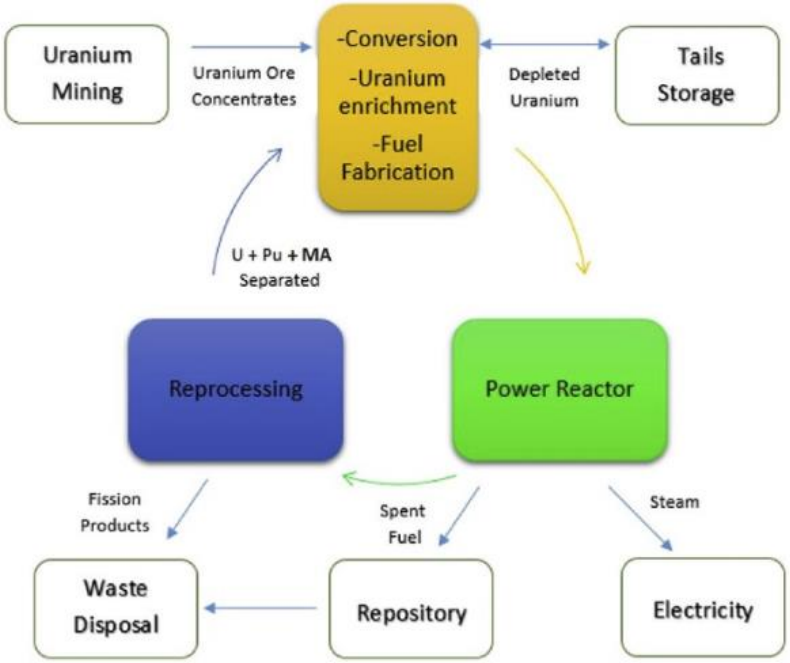


Figure 1-1: Example of connections between facilities [13]**.**

[1] MOX on Enriched Uranium Support

As can be stated, the irradiation process is a main encumbrance to be addressed pertaining to the final inventory for the nuclear cycle's back end. Hence, the scope of this thesis is indeed the irradiation model enhancement for ANICCA fuel cycle code.

1.2. The current irradiation model

The current irradiation model in charge of obtaining the final inventory is structured around solving the Bateman's transport equations by implementing CRAM (**C**hebyshev **R**ational **A**pproximation **M**ethod) [14], [15].

Although the first steps are commenced much earlier, in first place, it is required from a Monte Carlo depletion code (such as ALEPH2, in-house code from SCK-CEN [16]) to run and calculate the irradiation fuel variables which are later included in an ENDF format prebuilt library [13].

The library obtained via Monte Carlo calculations is then containing enough information to allow ANICCA to perform a burnup calculation in a single step. Simultaneously, other prebuilt libraries are required: those containing information about averaged flux and the averaged microscopic isotopic cross sections [12], [17].

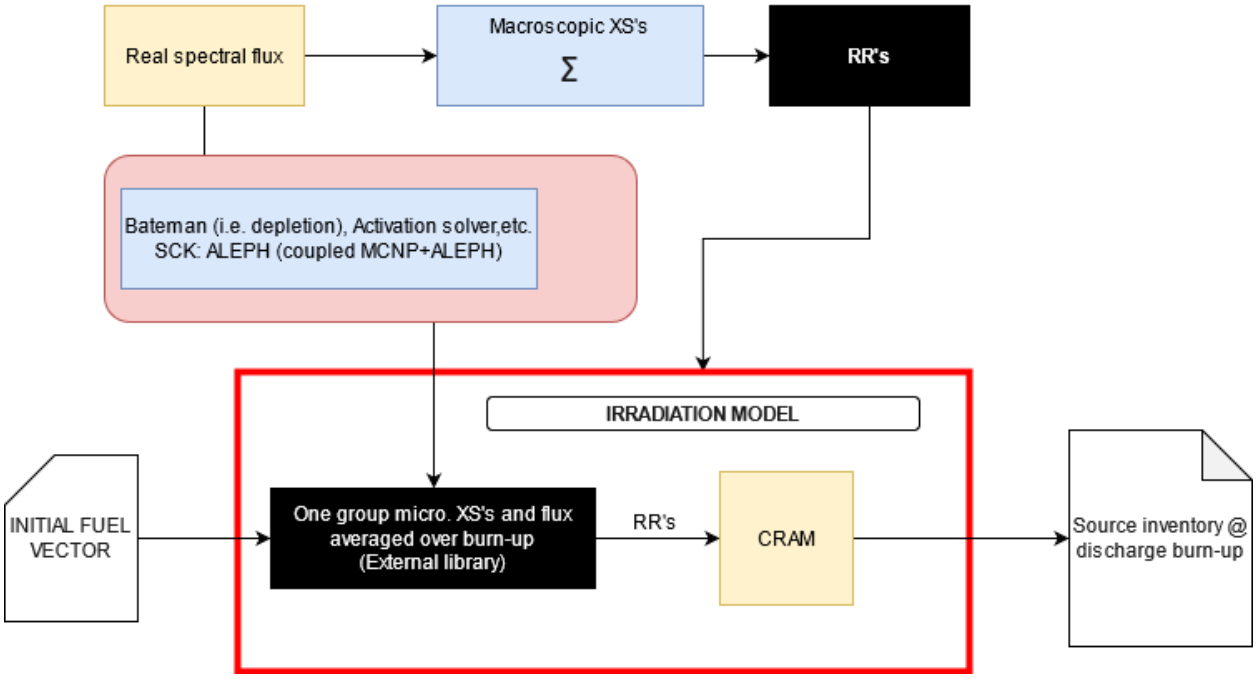


Figure 1-2: Flowchart of blocks involved in source inventory calculation through the current irradiation model.

These libraries are a rigid approach when compared to other methods given that a new library must be generated when the burnup of the scenario changes. Moreover, the contained cross sections are collapsed in one group and averaged over depletion time, which implies loss of information.

Thus, these aspects should draw the attention also regarding calculation efficiency:

- The need of requiring extra computational code for the fuel irradiation, which threatens code portability and modularity.
- The need of requiring extra libraries, which threatens the code simplicity and portability.
- To a lesser extent, the calculation time and resources invested in Bateman depletion calculations.
- The availability of trackable isotopes during burning is contingent upon the existence of those isotopes in the prebuilt library.

Special attention should be paid to the computational time issue. Yet being the amount of time required for a single calculation not extensive, the total computational time can easily rise due to the fact of stacking several steps for a long run scenario with many different reactors. In this case, computational time starts augmenting in a quadratic rise, in which time exigencies have soon reached non admissible computational costs.

1.3. Irradiation model enhancement proposal

Focusing on the irradiation model, an approach to enhance the current model is addressed in this thesis: to substitute the CRAM Bateman equations solver for a predictor based on machine learning techniques. This would allow almost instant forecasting and cause the computational time to reduce to the minimum achievable, whilst simplifying the in-built deployed network that supports the existing model.

Several machine learning algorithms can simplify calculations in a widespread range of applications. In this case, the chosen algorithm has been the Artificial Neural Networks (ANN) due to their capability of performing regression tasks while constantly lowering the error in predictions by continuous automated self-learning.

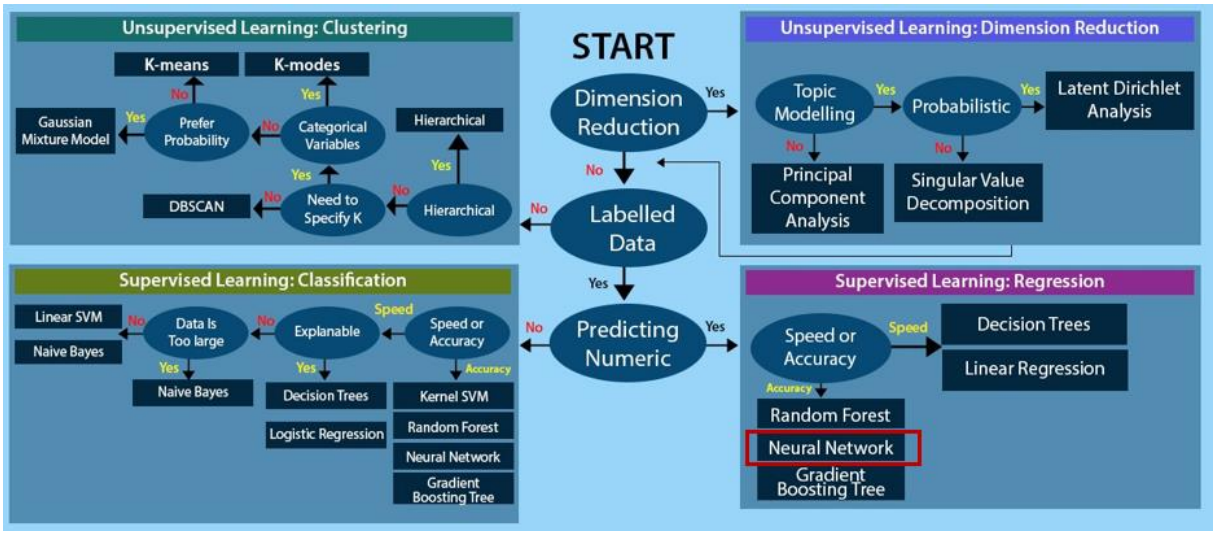


Figure 1-3: Machine learning algorithms decision chart [18].

The neural network has been chosen because accuracy is preferred to having high training speed (which differs from forecasting speed, but this will be discussed further into the

document). As in every machine learning algorithm, data is required to train the model so it can "learn" which features are the most important and how are they related in order to obtain an accurate prediction. Taking into account that fuel depletion codes can provide labelled data to train the model, the input of the ANN has been designed in the earlier steps for acquiring labelled data from other codes (mainly SERPENT2 depletion capabilities have been used) via data frames.

Given that the ANICCA code has been scripted in Python [17], it will be preferable that the ANN will be also built itself in the same code for the sake of integration. Besides, a huge support for this kind of algorithm implementation can be found in the Python environment, along with the data handling and analysis capabilities that the Python based modules offer (Keras, Tensorflow, Pandas, Numpy, etc.). All of these make Python the ideal environment to work with regarding machine and deep learning.

1.4. State of the art

Nowadays, Machine learning is widespread and integrated at many levels of data structure and many different sectors (finances, realty, production and manufacturing, IT, etc.), so the nuclear industry is no exception. In fact, several researches have been conducted in this field before, to be more precisely, neural networks have been implemented to cover a similar need for depletion calculations in the French CYCLUS fuel code [19] (check Figure 1-4).

The paper stated that well convergence was reached while maintaining lower and reasonable error margins. This objective seems to be feasible given this kind of solid feedbacks. A similar initiative was taken for the CLASS fuel cycle code. Also, similar outputs such as total neutron emission for SNF have been obtained via ANNs [20].

Moreover, and in a broader way, other machine learning algorithms have been implemented for different commitments, such as predicting the dynamic behavior of nuclear power plants [21]. Every examined report about the field has confirmed the feasibility of the question and the vast majority of them have shown a good performance regarding their different applications.

Nevertheless, full integration in a complete fuel cycle code is still expected to be enhanced. In order to do that, this topic pretends to do a further approach beyond the feasibility and trying to reach at least the early steps of a functional integration within the ANICCA code.

On the other hand, information required to design the neural network has been obtained from several reports about the relative importance from certain isotopes for SNF characterization [19], being classified in terms of neutron emission, heat emission, radiotoxicity, burnup indicators, etc.

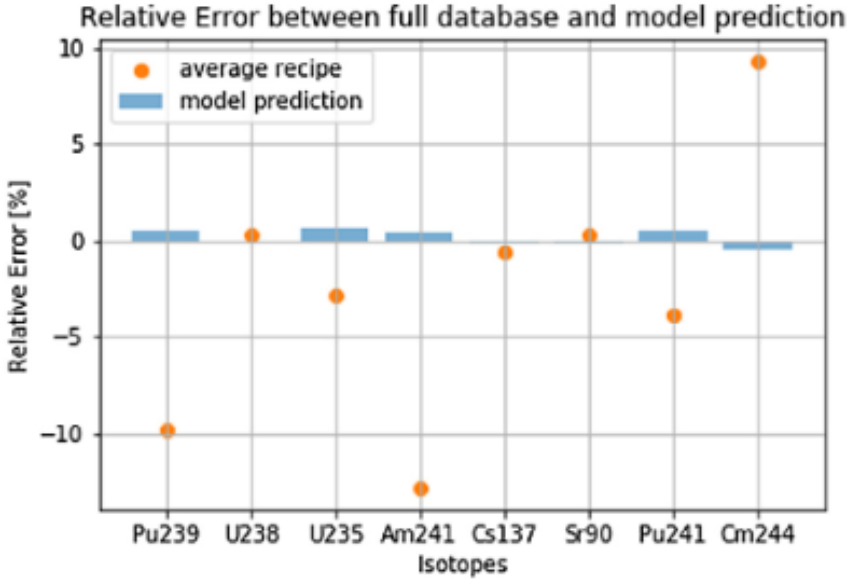


Figure 1-4: Prediction error against the original method for different isotopes in the CYCLUS paper [19]. *Please note the good fitting of the model given the error.*

1.5. Assumptions & Objectives

Given the aforementioned feedback, the following main objective for is:

To substitute the current irradiation model for a machine learning script (Neural Network based), to be able to forecast the final isotopic composition, given inputs as function of initial fuel composition as well as the final discharge burnup of interest.

To achieve this, the work baseline has been planned for the PWR technology, trying to forecast the different atomic densities for those more relevant isotopes involved in final processing and disposal of the different SNF. The base fuel types that were considered were MOX and conventional uranium fuel (UOX).

The input of the neural network will be both the initial enrichment (percentage of Pu or U-235 content depending on whether the fuel is MOX or UOX) and the desired discharged burnup, being these two variables known as the fuel vector.

The output will be the different atomic densities of the isotopes of interest in the fuel cycle analysis.

In order to achieve that, the following assumptions have been taken into account:

- Only thermal reactor models with PWR have been used as a pattern for arranging the labelled dataset.
- Other suitable features for the neural network to work with, such as decay heat or activity per isotope, have been dismissed for the sake of convenience.
- The dataset for the ANN will be generated via pin cell calculations using the continuous-energy multi-purpose three-dimensional Monte Carlo particle transport code SERPENT2.
- Neither helium nor other isolation gasses has been taken into account for modeling the gap to develop the training and testing datasets. Thus, the gap has been modeled as a void.

So, the main objective of this thesis can be sub-divided as follows:

1. To obtain a suitable dataset to train the initial neural network prototype.
2. To perform feature engineering on the different inputs given the feedback, so that the most sensitive of the features can be found.
3. To generate enough and high-quality data via SERPENT2, so that definitive neural networks can be trained.
4. To calculate and understand the different uncertainties existing in the SERPENT2 output, so that possible biases in the ANN can be analyzed.
5. To optimize the Neural Network, lowering the prediction error as low as possible.
6. To integrate the Neural Network trained model within the fuel cycle code ANICCA.
7. To run a whole fuel cycle in the new modified version of ANICCA.
8. To benchmark and test the forecasted inventory against the original data.
9. To assess the uncertainties in the model.

1.6. Document structure

Given the iterative nature of neural networks design, this document is structured in its further extension in a similar arrangement of the different processes that led to the composition of the definitive model. At the same time, the motivation of the different steps is commented. This is due to the very specific nature of the neural network algorithms: each problem and its dedicated neural network are very fitted one into each other, so the lack of flexibility is a main drawback when it pertains to elaborate a general-purpose neural network design process.

Also, knowing that machine learning technologies are perhaps still far away to be considered as a standard on the nuclear sector, in the next chapter, which comprises the ***Methodology***, a subchapter of neural networks' theory fundamentals will be introduced, as well as the descriptive stepwise process that was followed to model the most specific parameters and characteristics of the ANN.

Besides, the feature engineering (which is the procedure of picking, handling, and altering raw data into features which are employed in supervised learning) that has been carried away is also commented so the whole process could be easily followed. Finally, the isolation of the isotopes which are targets of interest is also discussed, taking into account their different nature and influence on the SNF final life analysis.

In the next chapter, the ***Results***, the different evaluations of the work done are shown, leading to the discussion chapter, where they will be synthetized into the main conclusions of the work along with the discussion of the possible impacts for this line of work. This fact shall be commented in the ***Work Impact assessment*** section, which also will be summarized in the ***Discussion*** chapter.

Finally, and to fulfill the engineering exigencies given the nature of this work, the budget, project organizational documentation along with a Gantt's diagram, as well as a relation of the different resources employed in the development of the work will also be presented.

2. METHODOLOGY

In the present chapter the whole methodology is presented. Firstly, it introduces the theoretical baseline on which the work has been built, followed by the different steps of the iterative process that has led to the design of the final modification for the irradiation model of ANICCA. A revision of the different methods available for auditing the build of the ANN is also included.

2.1. Neural Networks' theory fundamentals

An artificial neural network or ANN is a set of algorithms that emulates the human brain connections via mathematical modeling, lying the potential of this sort of method in them being capable of recognizing underlying patterns and relations within a set of data.

To achieve this, the neural network is fed with a vector input, which comprises the independent variables of the problem, and a vector output (which are the target data which is going to be forecasted, this means, the dependent variables). Thus, it can be said that the ANN needs a previous amount of data before initiating any prediction task.

Then, the ANN working principle is based on obtaining mathematical relations between the input vector and the output vector, so it is important to note that feature engineering over both of vectors will be an important aspect to consider (this means, to carefully choose the inputs so the forecasting could be efficient in terms of time and accurate in terms of error).

To illustrate how these mathematical relations are built, a conceptual layout of the topology of ANNs is shown in Figure 2-1:

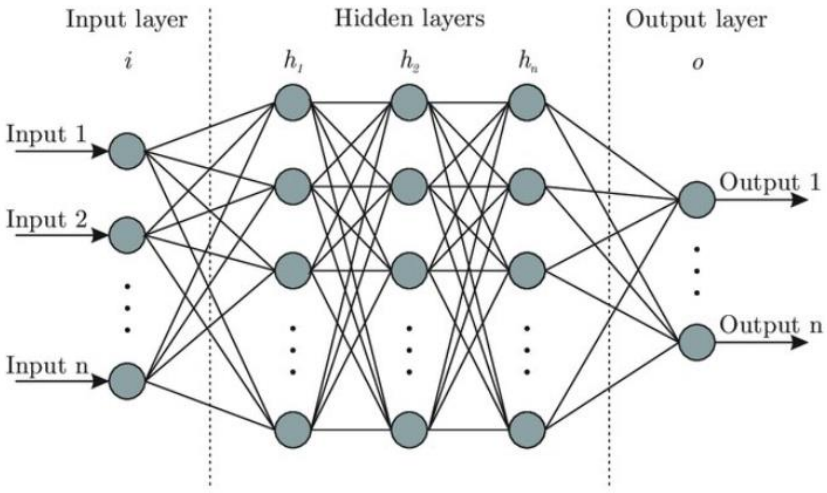


Figure 2-1: General layout for a neural network representation [22].

In the structure can be noticed three different objects: the neurons, the layers and the connections.

Every neural network have at least three kinds of layers: the input layer (***i***), which where the inputs are connected, the output layer (***o***), that limits the final end of the neural network and provide the outputs with the different predicted variables, and the hidden layers ($\mathbf{h}_i$), which are located in between the input and the output layer, the hidden layers transport the information from the input to the output being this information manipulated in the meantime. It is necessary to have at least one input layer and one output layer for an ANN to exist. Also, the number of hidden layers is very dependent on the kind of problem, the nature of the data, and the complexity of the relation between each data. Besides, and if conveniently, an ANN can have more than one input or output layer, this will depend on the kind of problem to be solved. Every layer is formed by neurons regardless of its position, neurons are the nodes represented in the previous image as gray dots, these nodes are an approach to a real-life human neuron. It is important to remark that no fixed number of neurons exists for each layer (except for the I/O layers, which number of neurons shall match with the dimensions of the input and output vectors respectively) and not all the layers need to have the same number of neurons.

It can be seen that the neurons belonging to a layer are linked to all the neurons existing in both the next and previous layer, this way every neuron is indirectly linked to each other. The connections are a conceptual representation of a weight ($\mathbf{w}_{ij}$), these parameters weight the connection from a neuron in a previous layer to another in the next layer, so by taking into account all the connections (or weights) that are fed into a neuron, it can be estimated the relative contribution that the information carried by those connections has on that neuron.

The neurons are the last and most complex part of a neural network. Their main function is to weight and add the different connections in its inputs (coming from a previous layer) and, when it is needed, activate its outputs to send the sum trough them. It is important to remark that the neuron always sends the same copy of the sum to every output it has.

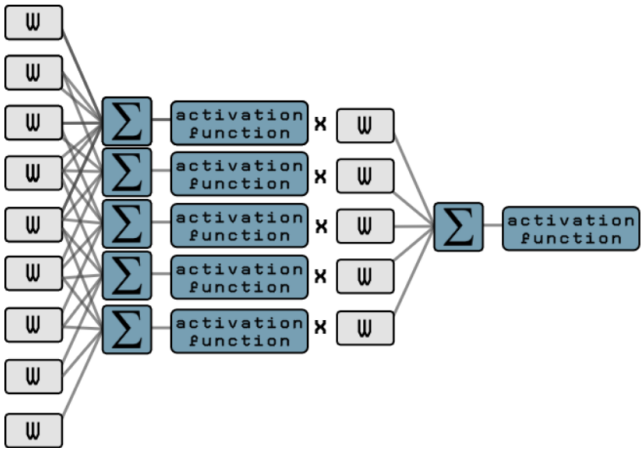


Figure 2-2: Mathematical representation of a six neuron layout [23].

It can be appreciated in Figure 2-2 how five neurons (blue colored) weight the different connections (gray colored) that have nine different weights (or nine different connection each), then those five neurons are connected to another layer containing one neuron, this latter, again weights the connections from the five neurons of the middle layer. The neuron comprises both the summation function and the activation function.

In short terms, an artificial neural network can be seen from the outside as a black box, which can predict an output Y when it is provided an input X and if it has been previously trained with a set of (X,Y) related data. Nevertheless, to understand how they are built in order to reduce the flaws in predictions requires more insight about their working principle.

ANNs are employed in many applications such as classification problems, image recognition via convolutional neural networks and so on, but in this case, only the ANN-based regression model is relevant for the thesis scope.

2.1.1. Activation in neural networks

One relevant aspect of the neuron working principle is the concept of neuron activation. When a neuron has summed the input weights it will not emit an output if the neuron is not activated.

This activation is regulated by a so-called activation function which governs all the neurons that belong to the same layer. The activation functions are rules that activate certain neurons that have crossed a threshold, which results in not having all the neurons activated simultaneously, thus, this is done so non-linearity can be introduced in the model. This activation is regulated by a so-called activation function.

To enlighten the utility of these activation functions, it should be convenient to state the Universal Approximation Theorem [24],[25]:

"***A feed-forward network constructed of artificial neurons can approximate arbitrary well real-valued continuous function on compact subset of*** $\mathbb{R}^n$**."**

Also, knowing that the neurons are basically handling a linear combination of n variables with n weights, and knowing that the linear combination of linear functions must be linear, it can be stated that no non-linear regressions could be executed if there were only linear functions between the input and the output layer. Hence the need for activation functions.

There are plenty activation functions that can be used depending on the task to solve, in the imaging classification problems it is often used enhanced variations of the sigmoid functions such as the Softmax function or solving categorical classification could require a binary activation function [26]. But in the case of regression, it will strongly depend on data scaling and normalization. To illustrate this, an example of the most employed function for regression is shown in Figure 2-3: the ReLU function or **Re**ctified **L**inear **U**nit.

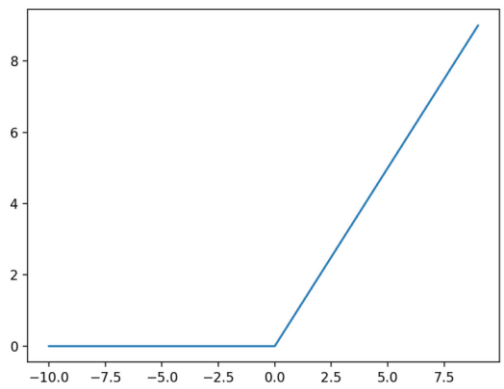


Figure 2-3: ReLU activation function.

The ReLU function provides null values per each negative value given whilst being linear in the positive plane of the input so the negative outputs from certain neurons could be avoided. Nevertheless, there are many aspects to consider when it is needed to select the activation functions, among these, the most important are [27]:

- Category of problem (image recognition, classification, linear regression, non-linear regression).
- Position of layer (e.g., if zero values are not wanted in the predictions, ReLU functions should be avoided in the hidden layer because it tends to initialize every negative as zero, see Fig 1-6.)
- Normalization of data (if the range in which the data are normalized does not correctly fit the activation function threshold, it is expected for the neural network to increase its error, e.g., ReLU does not work with negative values, so scaling the data from -1 to 1 will not be successful).

2.1.2. Working principle

In order to properly train a neural network, two steps must be completed: forward propagation and back propagation [28].

In the first step the information travels inside the ANN from the input layer to the output layer, first, each information unit is weighted through the connections with a pre-initialized weight set. In fact, the initialization of these weights is a tunable and critical parameter to be aware of so high quality predictions can be performed. Then every neuron sums its inputs and it will be triggered if the activation function allows it, otherwise every input will be cero. Nevertheless, a bias can be also inserted by a neuron when a null value reaches it [29].

The information is then propagated through the hidden layers until it reaches the output layer(s). This process is repeated with different batches of data, which are bundled according to the amount of available data in the training dataset and the accuracy or time criteria requirements that the design must meet.

The data, in the shape defined by the batch size will pass through the NN as many times as the designer states via another parameter, which is called the epoch; both batch size and epochs are part of the hyperparameters family.

Up to this point, the only affirmation that can be made is that, indeed, the neural network topology working altogether is capable of modifying the outputs with many different parameters (if every connection is portrayed as a weight to be tune, is easy to understand that the number of the parameters which affects the final result is proportional to the size and number of layers). However, the forward propagation mechanism is not able to check and correct the deviation between the tentative values and the real values, which is the reason why the back propagation algorithm is implemented.

2.1.3. Learning principle: Back propagation

Once the first epoch has concluded and the first hinted value set for the weights has been written, the neural network needs the back propagation algorithm so the different adjustments to be made can travel through the hidden layers from the output layers to the input layers of the ANN. With this, the neural network can adjust the weights basing its tuning on the expected output from the training data, hence the name of back propagation [26], [28].

In first place, the loss function (or cost function, which in regression could be represented by an error function, i.e., mean squared error, etc.) between the predicted output and the actual values from the training set is calculated. Then the gradient of the loss function is calculated per each output by the back propagation algorithm.

To update the weights a second algorithm is called, the Gradient Descent [29] (which will be introduced in the next point). This algorithm is firstly focused on the activation outputs from the given output nodes (neurons in the output(s) layer(s)), so the algorithm notices which values should increase (because they are directly interacting with a correctly predicted value) or rather decrease (to decrease the importance of this weights given that they are producing some incorrect values when tested against the actual values for the output). When that is done, the task is passed to the back propagation algorithm.

Knowing then that the values from these output nodes come from the weighted sum of the previous connections being multiplied by the output of the previous layer, it is clear that if the output values should be changed, the connection weights between the hidden layer and the output layer should be updated. Then a change in the activation functions from the previous layer is indirectly forced by the back propagation algorithm, which jumps backwards and again updating the weights of every connection between the layers.

This process is sustained until the input layer connections are reached, because it is not desirable to change the activation values in the input layers since this contains the actual input data.

It must be stated that the proportion on which every parameter is changed in relation to the other may be higher or lower depending on how much effect the update is going to have on the network as a whole to lower the loss. The updated values that remain for each of the corresponding weights are actually the corresponding derivatives of the loss function with respect to each of these individual weights.

The aforementioned process that implies just one input is repeated for all input for each batch provided to the network, and the resulting updates to the weights and the network are going to be the average updates that are calculated for each individual input. These averaged results for each weight are indeed the corresponding gradients of the given loss function with respect to each weight.

2.1.4. Learning verification. Lowering the error: Gradient Descent

To assure that the back propagation algorithm leads to the correct result update, the error must be lowered following a strategy which is based on the Gradient Descent optimization algorithm. Hence, the back propagation is proof tested against oscillation (which will lead to a divergent sequence with the updatable parameters increasing their error instead of reducing it) [29].

The main idea of the algorithm is to minimize the loss function, which represents the estimative error. This loss function could be aiming for probabilistic losses, i.e., binary cross entropy, Poisson function, etc. or rather for regression losses i.e., mean squared error, Huber function, Log-Cosh function, etc.

Therefore, gradient descent is an iterative optimization algorithm which finds a loss function's local minimum by first computing the gradient (first derivative of function at that spot) and secondly, stepping towards the opposite direction of the gradient (which leads to a function's module minimization). The algorithm could be scripted in general terms as follows [30]:

$$repeat\ until\ convergence\ \{$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(for\ j = 1\ and\ j = 0)\}$$

Being $\boldsymbol{J(\theta_0, \theta_1)}$ the loss function to minimize, $\boldsymbol{\theta_0}$ and $\boldsymbol{\theta_1}$ the parameters of the loss function and being $\boldsymbol{\alpha}$ known as learning rate, which is a tuning parameter crucial for the optimization process and devoted to control the step length which directly affects effectivity in both the training process and the convergence of the algorithm. Having lower learning rates will lead to an increased demand on computational time whilst having high efficiency on loss function minimization. Having higher learning rates will lead to the opposite situation having more bearable computational times in lieu of efficiency. If the learning rate is too high, the algorithm will diverge, which is known as overshooting, so this parameter is tuned attending to a tradeoff between both aspects of neural network design.

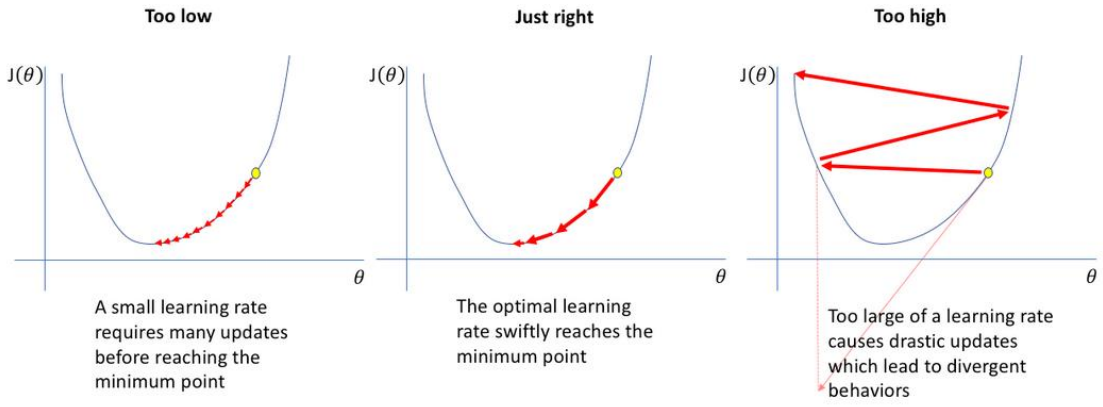


Figure 2-4: Different effects of learning rate setting [31].

The process followed can easily be interpreted by glancing at a contour plot of the loss (or cost (function) in Figure 2-5:

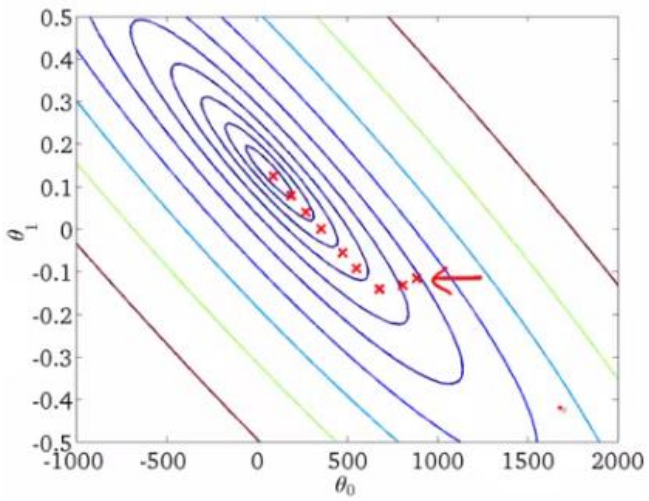


Figure 2-5: Contour plot of the loss function, the marked spots represent the consecutive values where the algorithm is pointing at [30].

In the graph of the Figure 2-5 it can be stated that the response value increases away from the center having the same value along the different level curves; the red crosses indicate the steps taken by the algorithm.

It also shall be remarked that the existence of local minima could bias the algorithm by having it reaching convergence in a minimum that is not the lowest of the minima set. This strongly depends on the initial point taken (initial parameters) and on the learning rate, so it is convenient to invest time in finding and understanding these relations to optimize the neural network prior to its design. A way to avoid this problem and others, like having the algorithm settled in a saddle point (minimum in one direction and maximum for another), is using the Stochastic Gradient Descent.

In the stochastic gradient descent, instead of using the stepping technique for computing the gradient of the loss function, a step is taken by computing the gradient loss of only one random sample. At each step the gradient, if a loss function different from the actual loss function is taken, may point to a slightly different direction than the gradient of all the-whole-samples direction has. This translates into constant change in the randomly sampled data point, so being in an iteration in which the algorithm settled in a local minimum, in the next iteration the next sample might be different, which keeps moving the direction until the global minimum is reached.

Nevertheless, this is not a very feasible option regarding the computational cost, that is because the gradient of the individual losses supports parallel calculation whereas sequential calculation is required in case of the stochastic option.

Then, the tradeoff is granted by a balancing act: the batch concept, which was introduced in section 2.1.2, where the data points are processed in groups instead of calculating the parameters for the loss function one by one, and then the stochasticity to avoid stagnation at local minima is provided.

These connections between the learning rate tradeoff and the local minima issue sets the ideal scenario to revisit the learning rate concept. Traditionally, the training is done following the theoretical-based procedures(i.e., fixed number of iterations, but recent state-of-the-art literature is shaping the concept of early stopping, which is a technique based on bringing the training process to a halt after a certain number of iterations have been done whilst the loss is not improving. This, if used along a decaying learning rate, where the learning rate is decayed each time the loss fails to improve after a given number of iterations [32].

This supposes a wide movement range in the earlier steps that account for the local minima problem and then, a step size reduction increases the accuracy in the last steps.

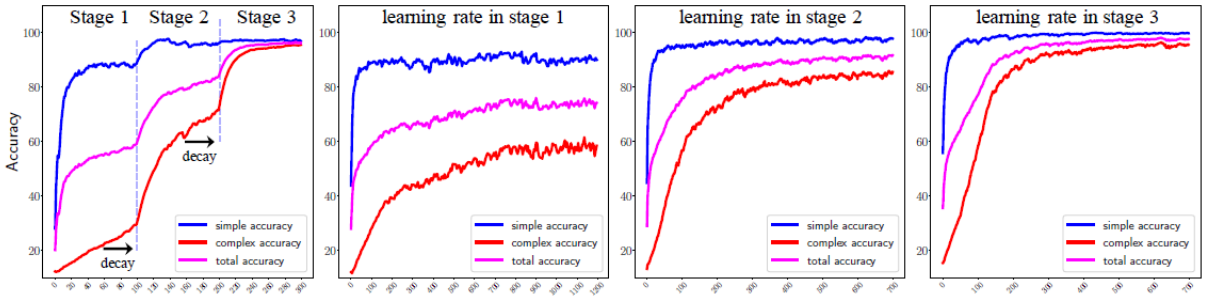


Figure 2-6: From left to right: training with decay learning rate; training with constant learning rates for stage 1,2 and 3, the X axis shows the epoch number. Note that the difference between the epochs needed for reaching high accuracy levels in both cases is remarkable [32].

2.1.5. Performance: Overfitting and Underfitting

The generalization concept refers to how correct are the concepts learned by the neural network when the model is applied to specific data that has not been seen by the model during the learning process. It is a concept related to the performance of any machine learning model, and it deals with the two major problems regarding performance: overfitting and underfitting [33].

Overfitting is the effect by which the model learns both the detail and noise in the training set, and this impacts the performance when tested against unseen data, i.e. the noise and fluctuations are learned as something of importance for the features, so having different noise levels in new data will cause a mismatch between new and seen data.

Underfitting is the opposite effect, in this case the model is unsuitable because it is unable to either generalize new data or to model the training set. So, the ideal range is a tradeoff between the two issues.

This encumbrance can be addressed by producing a validation data set (which is a subset of the training data spared from being used for training), this data set is then used for testing the model performance against unseen data.

Several variations of this technique are known: validation (splitting the whole data set into training set and validation set) and K-Fold cross validation (dividing data sets into K number of subsets, and using them in different number between epochs resulting in having different testing data set per each epoch) [34].

2.2. Preliminary research

In the initial steps of the project, a 279 isotopes database was taken as reference to understand the main different characteristics to be taken into account when designing the neural network. This database came from Uppsala university in Sweden and was retrieved from Mendeley Data [35]. It consists of a table-structured comma separated value file which contains filtered data, the data was generated by computer simulations with the SERPENT 2.1.28 Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code [36].

The data contains a wide variety of atomic densities for 279 nuclides, these densities were collected for a wide range of initial enrichments (IE) or initial plutonium content (IPC), discharge burnup (BU) and cooling time (CT).

The simulation consisted of a single pin cell burnup calculation with the parameters depicted in Table 2-1:

Table 2-1: Parameters of the model in SERPENT2 for the 279 isotopes database [35].

Fuel pellet radius (cm)	0.41	**Coolant density (g/cm^3)**	0.75
Clad inner radius (cm)	0.42	**Coolant temperature (K)**	600
Clad outer radius (cm)	0.48	**Boundary conditions**	Reflective (set bc 2)
Pitch between pins (cm)	1.26	**Number of histories per generation**	5000
Fuel material	Low enriched UO_2 MOX	**Number of active generations**	100
Fuel density (g/cm^3)	10.5	**Number of inactive generations**	10
Fuel temperature (K)	1500	**Power density(kW/g)**	$27.397 \cdot 10^{-3}$
Cladding material	Natural Zirconium	**Neutron cross section library**	JEFF 3.1
Cladding density (g/cm^3)	6.52	**Decay and fission Yield Data**	ENDF-B-VI-8
Cladding temperature (K)	900	**Operational history**	10 MWd/kgHM burnup each year (365 days) calculated in 0.5 MWd/kgHM steps. After every 10 MWd/kgHM burnup period, a 30 day downtime (set powdens 0) is included. Burnup goes up to 70 MWd/kgHM (i.e. 7 years of operation)
Coolant material	Pressurized water		

The MOX fuel compositions were calculated with the following fuel vectors presented in Table 2-2.

Table 2-2: Fuel vectors for MOX calculation in reference database [35].

Plutonium vector (w%)		Uranium vector (w%)	
Pu-238	2.5	U-234	0.0012
Pu-239	54.7	U-235	0.25
Pu-240	26.1	U-238	99.7488
Pu-241	9.5		
Pu-242	7.2		

Summarizing all the variables, there are in total 131 different burnup values, 131 different cooling time values and 46 initial enrichment values for LEU fuel (from 1.5% to 6.0% in increments of 0.1%) and 31 different plutonium content values for MOX fuel (from 4.0% to 10.0% in steps of 0.2%); which means 789,406 grid points for UOX and 531,991 grid points for MOX fuel.

These grid points are arranged in a 2D array-like table with the columns classifiable as features[2] being: Burnup ('BU'), Cooling Time ('CT'), Initial enrichment ('IE') (enrichment for UOX and Pu percentage for MOX), Fuel Type ('fuelType') (UOX or MOX), Spontaneous Fission Rates, Photon Emission Rate, Activity and Decay Heat. The rest of columns (which represent the target[3]) are the 279 nuclides: H1, H2, … Be9, B10, etc. This huge amount of data (7.7 GB) cannot be handled by using conventional tools like spreadsheets, so in this case the Pandas Python-built data analysis tool has been used, together with Matplotlib library to introduce visualization capabilities for data curation and diagnosis.

2.2.1. Data curation

The first step was to curate the data by dropping the columns that are not needed in the neural network ecosystem, this means, only the BU, CT, IE, fuelType and nuclides' columns remained, reducing the size from 1,321,397 x 287 to 1,321,397 x 283.

Then, for the sake of convenience, the feature 'fuelType' was exploited to split the database into two subsets: MOX and UOX. This reduces considerably the loading times when processing .csv files and dataframe operations, and it is also reasonable from the neural network's perspective, because both fuel types can be handled separately. Finally, the fuelType column was also dropped providing that this feature is not used anymore.

The next step consisted in plotting the atomic density against rather the initial enrichment or the burnup, in order to observe different patterns between features and targets. In some cases, two different populations can be noticed, Xe-135 is a good example of this trend as can be seen in Figure 2-7:

[2] Features are every measurable property which serves as an explanatory variable for the data prediction.
[3] Targets are the values to be predicted, i.e., the 279 nuclides.

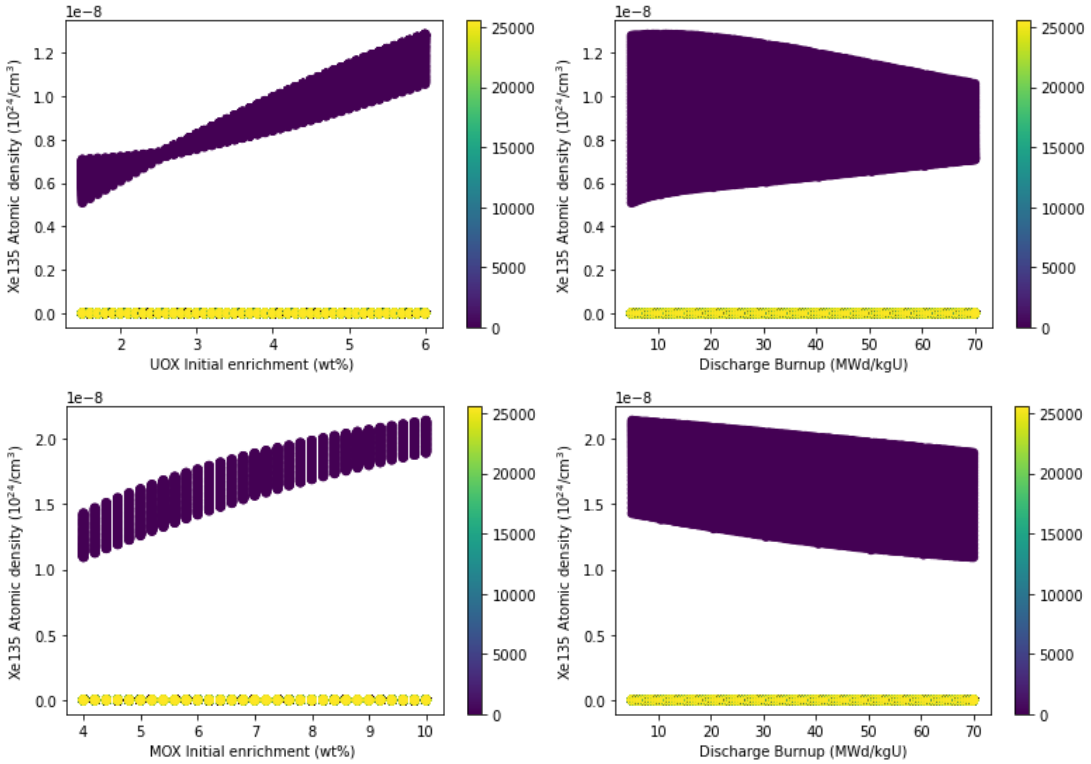


Figure 2-7: Evolution of the atomic density with the initial enrichment and the burnup. The color function is the cooling time (in days), so it can be noted how it affects the atomic densities.

It is clear than the clustering of several points seems to be correlated with the cooling time, this is an obvious thing to expect providing that, in certain short half-life isotopes, the depletion is noticeable in relatively short time periods. This is the case of Xe-135, which has a half-life of 9.2 hours.

This fact allows the possibility of using the feature of cooling time to filtrate data, and this seems reasonable given that no cooling times are needed in the predictions required for the thesis' objective. Therefore, the data was curated by only picking rows with CT==0, this means that only a 30 days without irradiation period was included (according to the author of the database, after 10 MWd/kgHM of burnup, a 30 day downtime period was included, which seems useful to simulate reloading stages in PWR cores). Once the selected rows were appended into a new dataframe, the cooling time row was also dropped for the sake of simplicity.

Also, in relation with the concern about the suitability of dispensing with a feature, an anticipated neural network was launched for another secondary database, this second database was obtained from the work of SCK-CEN researchers [37], and gathered the total neutron emission of SNF for different mechanisms. Once it was trained and launched for new predictions, the results showed that the points on the actual vs. predicted plot tended to cluster along the y=x diagonal.

So, providing that neutron emission observables are of major importance, along with the lack of necessity of including different from null values for the cooling time feature, it can be stated that the assumption of dropping it seems solid enough.

The clustering effect with which the cooling time feature distorts the forecasting can be observed in Figure 2-8.

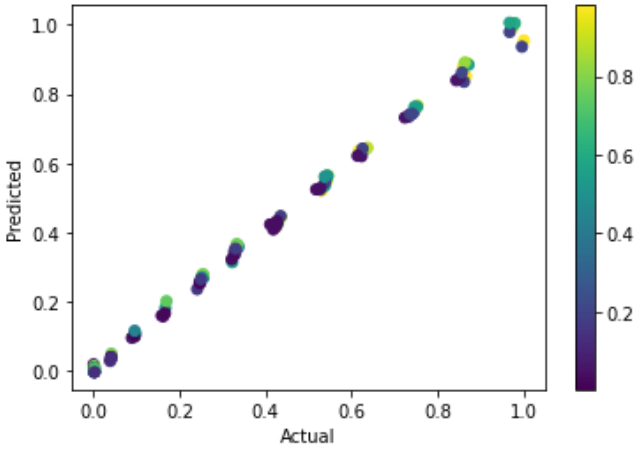


Figure 2-8: Predicted vs. Actual plot for neutron emission rates. The color represents different cooling times.

Note that the spots with a certain cooling time value tend to draw a line with a different slope or trend, which causes a clustering effect and adds a bias to the predicted values.

2.2.2. Neural network prototype

One of the early steps of the work was to test the project feasibility, so to achieve that, two basic neural networks were set up, one per each fuel type. This neural network worked with a two-dimensional input (initial enrichment and discharge burnup), and delivered a 279-D vector consisting of the different atomic densities to be predicted for each isotope of the previously curated dataset.

The chosen environment for setting the neural network has been the Keras API, which runs over Tensorflow. Once again, Pandas and Matplotlib have been imported to provide data managing and visualization capabilities.

2.2.2.1. Training and test sets arrangement

The first step consisted in obtaining two pairs of subsets from both the MOX and UOX databases,. This operation was carried away by using a random selection script which split the whole fuel database into training set and test set with a 80/20 proportion.

Then, the data were scaled between 0 and 1 by using the '*MinMaxScaler*' Scikit learn preprocessing function [38] , which assigns 0 to the minimum of a Pandas column (IE, BU, atomic densities, …) and 1 to the maximum of each column. This is done since, if the theoretical fundamentals of neural networks are revisited, it is obvious to state that the data must be in the range of the activation functions that are used, otherwise it will cause the activation functions to saturate and the neural network to malfunction. Therefore, given that most activation functions suitable for regression problems are in the (0,1) range, it seems reasonable to do so.

2.2.2.2. Neural Network prototype: Architecture and Hyperparameters

The prototype was built without following any hints from an optimizer and the hyperparameters were manually tuned (i.e., number of layers, number of neurons per layer, epochs, batch size, validation split). The loss function was chosen as the metric to monitor the performance (mean squared error, a traditional loss function used for regression problems).

In the UOX network, a minimum training loss of up to 0.0013 was obtained. The validation losses reached 0.002 and, when tested against the test data set, it returned a mean squared error value of 0.00155. Also, the number of epochs (which was set to 30) was obtained from the observation of the loss descent through the training process:

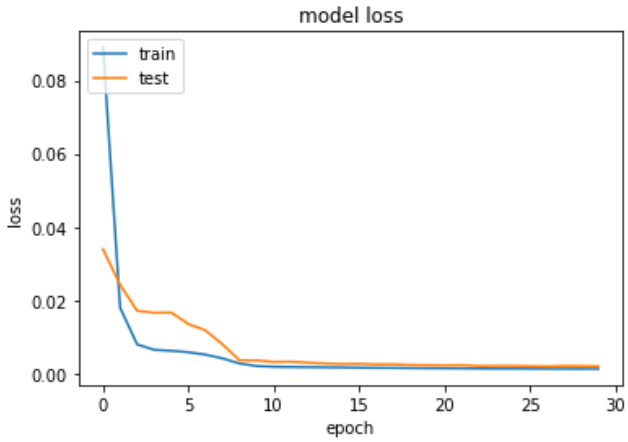


Figure 2-9: Test and training losses for UOX neural network along the progression of epochs.

In the MOX network, training losses up to 0.0029, validation losses (for the validation split) up to 0.0031, and when tested against the test data, losses of 0.00334 were achieved. Also, to set the number of epochs, the loss evolution through the epochs was monitored:

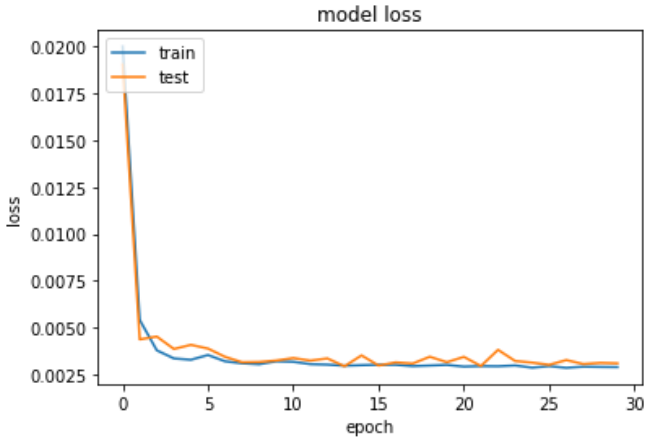


Figure 2-10: Test and training losses for MOX neural network along the progression of epochs.

Consequently, it resulted that setting the number of epochs that minimized the losses for both testing and training sets turned out to be about 30. The architecture of both networks can be observed below:

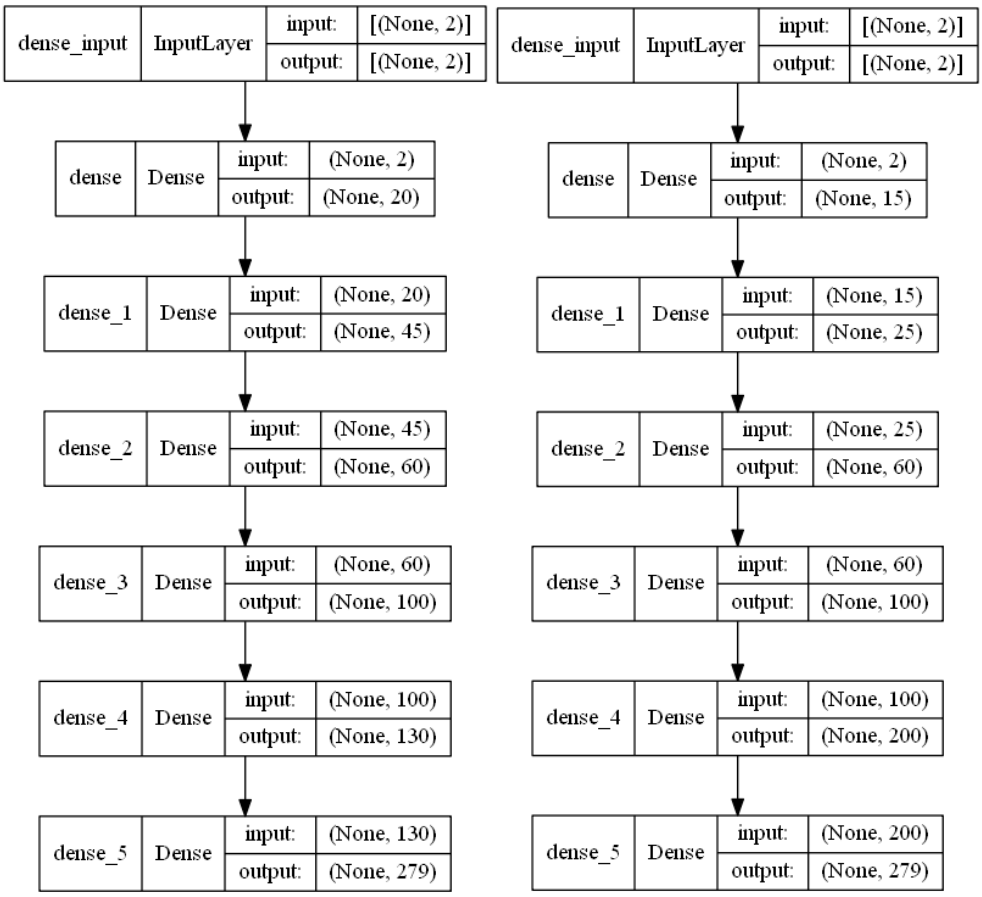


Figure 2-11: The model on the left is for UOX fuel, the scheme on the right represents the MOX fuel model. Dense means that every neuron in the layer is connected to every layer in both the previous and the next layer. The dimension (None, x) states that no length (number of rows) was defined, and x is the number of neurons connected to each layer.

Table 2-3: Summary of both neural networks' architecture.

	UOX ANN	MOX ANN
Number of layers (activation function)	6 (selu[4]) + 1(linear[5], output layer)	
Kernel initializer	Normal[6]	
Optimizer	Adam (based on Stochastic gradient descent)	
Epochs	30	
Training batch size	64	5
Testing Batch size	128	
Learning rate	0.001	

[4] Scaled Exponential Linear Unit, a variation of ReLU, used for preserving variance of datasets.
[5] Empirically, the typical function to be used in the output layer for regression models.
[6] Neuronal weights are initialized following a normal distribution.

2.2.3. Results and post processing

The technique followed to estimate the prediction quality relied on the actual vs. predicted plot concept; each nuclide was tested in this plot and then a linear regression study was conducted to obtain both R squared and MSE. For the sake of simplicity, the results are shown in statistical terms:

Table 2-4: Percentiles of mean squared error for UOX and MOX fuel.

UOX FUEL				
PERCENTILE	25	50	75	100
MSE	0.00028044	0.0004217	0.0007376	0.0401464
MOX FUEL				
PERCENTILE	25	50	75	100
MSE	0.0005166	0.00072441	0.00106773	0.06889585

The results can be ranked by using the R squared, then the best 12 estimated values in terms of R squared for each fuel type are:

Table 2-5: Ranking of the 12 best estimated values for MOX and UOX fuel.

UOX FUEL			MOX FUEL		
Isotope	R squared	MSE	Isotope	R squared	MSE
Ru-102	0.999833	0.000269	I-129	0.999697	0.00055
Sb-123	0.999833	0.00029	Cs-135	0.999697	0.000736
Sn-118	0.999828	0.000391	Eu-153	0.999662	0.000395
Cs-137	0.999827	6.99E-05	Ce-140	0.999635	0.000434
Cd-116	0.999794	0.000458	Ba-136	0.999612	0.000284
Sm-150	0.999779	4.78E-05	Ba-138	0.99961	0.000472
Sn-126	0.999776	0.000303	Sn-115	0.999604	0.000445
Se-78	0.999775	0.000139	Cm-244	0.999602	0.000413
Sm-154	0.999771	0.000306	Nd-146	0.9996	0.000385
Sn-117	0.999769	0.000354	Dy-164	0.999572	0.000281
Te-126	0.999768	0.000546	Kr-80	0.999549	0.000642
Te-130	0.999767	0.000261	Pd-107	0.999516	0.000571

At a glance, it can be stated that the ranking for both fuel types is not the same, so it seems reasonable to split the work into two different neural networks, or to add the feature “fuelType” back again to the dataset. However, this will transform the regression problem into a convoluted classification and regression problem, hence, an increasing of the complexity of the neural network is expected. So finally, remaining with two separated workflows was the chosen choice.

At the same time, the 12 worst R squared estimated values were withdrawn in order to understand the different bias found in the correlations between actual and predicted:

Table 2-6: Ranking of the 12 worst estimated values for MOX and UOX fuel.

UOX FUEL			MOX FUEL		
Isotope	R squared	MSE	Isotope	R squared	MSE
Xe-133	0.003317	0.040146	Pr-143	0.00139	0.052055
La-140	0.057473	0.031806	Ba-140	0.001883	0.050905
Nd-147	0.074238	0.029301	Xe-133	0.007272	0.043524
Ce-141	0.155711	0.035234	Ce-141	0.009313	0.065437
Ba-140	0.201916	0.024444	Zr-95	0.013679	0.068896
Pr-143	0.449601	0.016993	Y-91	0.015847	0.057239
I-131	0.457685	0.017792	Ru-103	0.024408	0.066846
Mo-99	0.508023	0.010441	Nd-147	0.029698	0.04269
Zr-95	0.59198	0.016829	Sr-89	0.030356	0.051495
Nb-95	0.616204	0.018102	La-140	0.033654	0.040879
Te-129m	0.724477	0.013387	I-131	0.037926	0.038255
Ru-103	0.779315	0.011386	Te-129m	0.04055	0.052186

To offer an idea of how many kinds of scattering patterns are, an array of plots of the four worst estimated nuclides per fuel is provided:

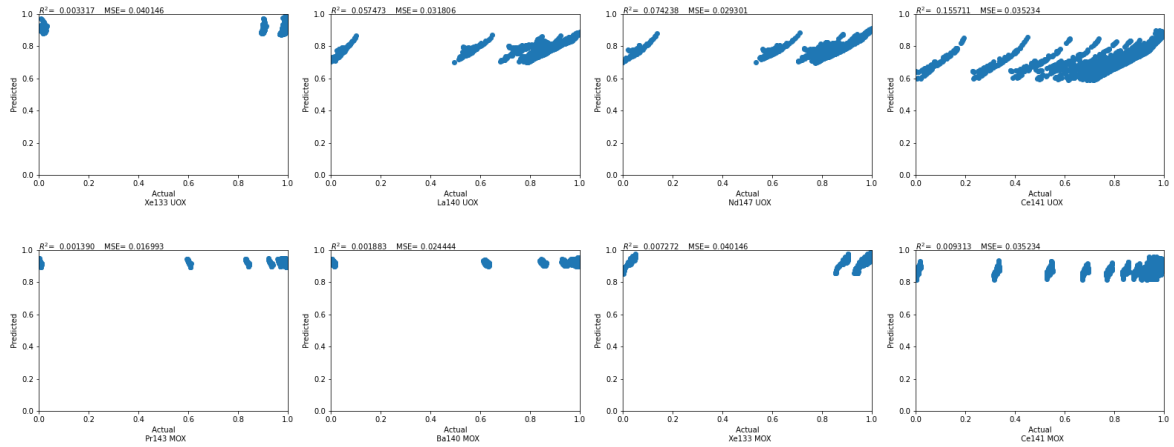


Figure 2-12: Scatter plots of actual vs. predicted of 8 of the worst predicted nuclides, note that no pattern can be found between the different points.

The points of the scatter plots are indeed randomly placed, with no linear or polynomial possible fitting. When an adjustment over the hyperparameters was tried in order to enhance the prediction over these nuclides, the rest have tended to be predicted even worse. Thus, it has been supposed that there are certain isotopes which cannot fit in the same neural network with a suitable accuracy given that or they are rather treated as a white noise (due to data availability over these nuclides or neural network internal operation) or their production paths are too complex for these isotopes to be treated in the same vector as the rest of the nuclides (the complexity of the data relations is inversely proportional to the forecasting accuracy).

2.3. Observables election criteria

Following the insight of the preliminary research, it was decided for the sake of efficiency to set the neural network span according to those nuclides that are relevant from both the fuel cycle and spent nuclear fuel characterization perspectives. Then, to gather the relevant nuclides an extensive bibliography was revised, nevertheless two main references were followed to provide the final database structure:

- For the spent nuclear fuel characterization, the nuclides were selected according to the 2018 JRC technical report: *Observables of interest for the characterization of Spent Nuclear Fuel* [39] .
- For the fuel cycle analysis, the nuclides were selected according to 2009 Idaho National Laboratory article: *Selection of Isotopes and Elements for Fuel Cycle Analysis* [40].

2.3.1. SNF characterization

Regarding SNF characterization, the effects of the different nuclides over the decay heat, the gamma and the neutron emissions have been considered.

Decay heat is an important aspect to take into account pertaining the interim storage and final disposal designs.

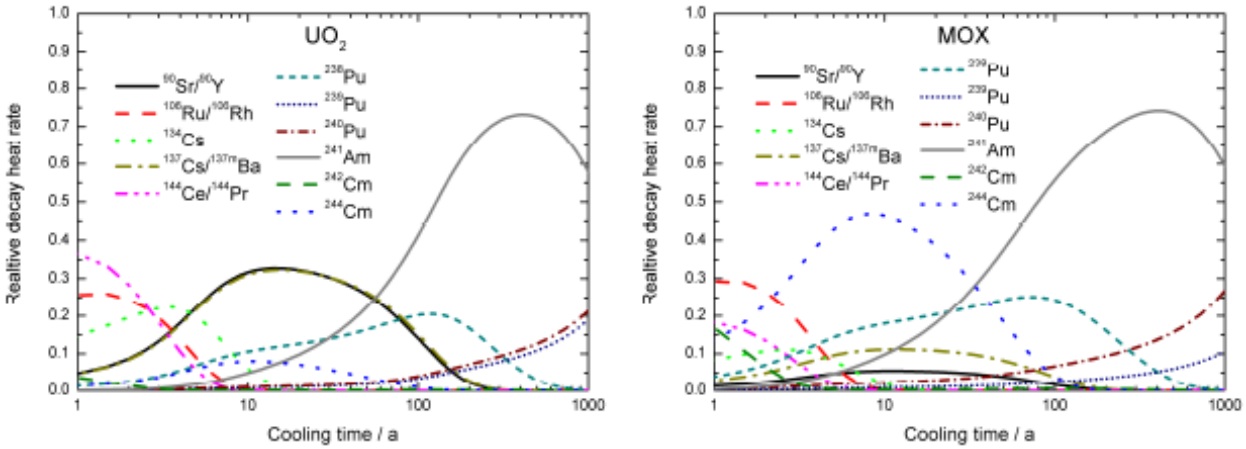


Figure 2-13: Relative contributions for the different studied nuclides studied in the JRC report [39].

In terms of photon emission, it should be noted that similar behavior can be found in both the MOX and traditional UOX spent nuclear fuel given that the gamma spectra for cooling times less than 100 years are majorly generated by the fission products, even though some differences are expected in fission products yields for neutron induced fission of U-235 and Pu-239/Pu-241.

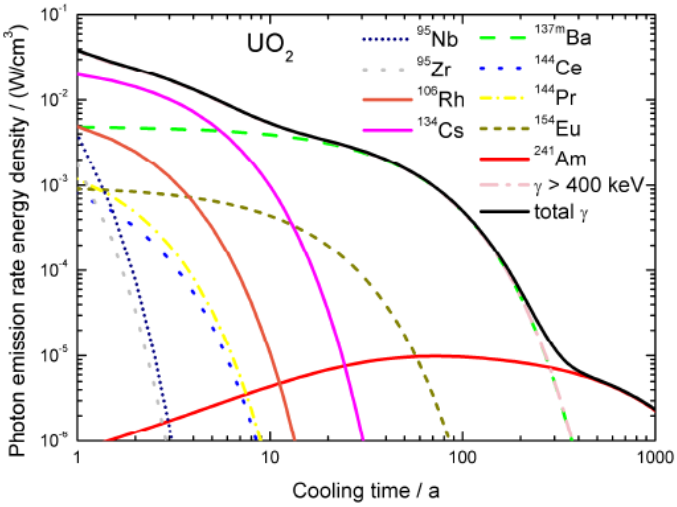


Figure 2-14: Gamma emission rate energy density contribution of certain nuclides as a function of SNF cooling time [39].

From the neutron emission perspective, isotopes in Figure 2-15 have been considered. However, not all the isotopes pose a contribution in the short run, such as Cm-242, which is only important for short cooling times:

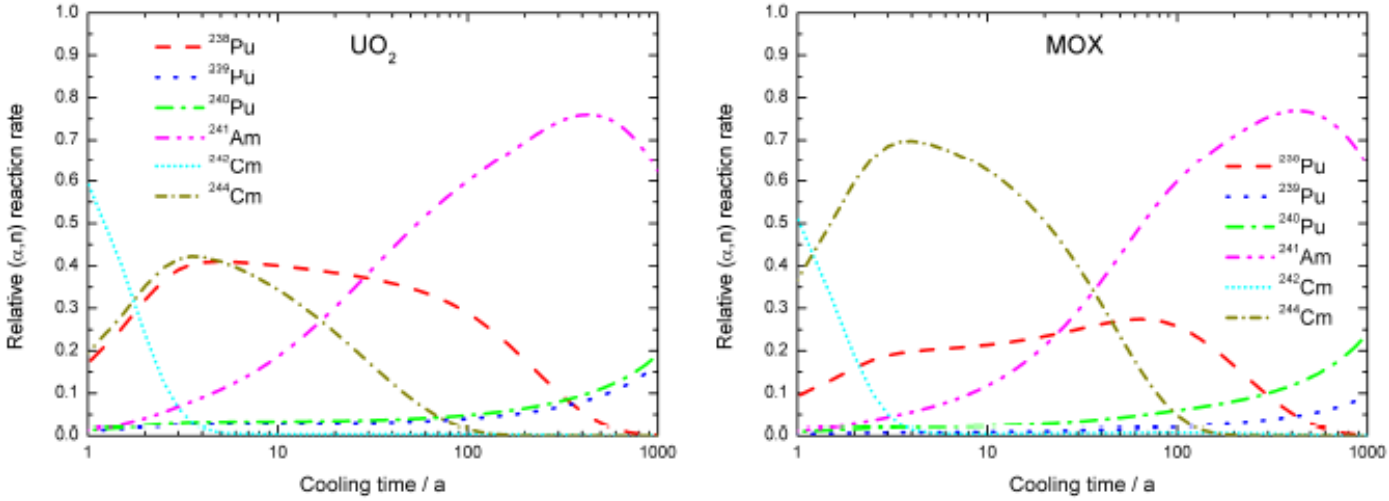


Figure 2-15: Relative contribution of plutonium (238,239,240), curium (242,244) and americium 241 to the total neutron emission [39].

2.3.2. Fuel cycle analysis

From the fuel cycle perspective, special attention must be paid to different key isotopes such as those belonging to the Group 1A/2A (Rb, Cs, Sr, Ba), noble gasses (Kr, Xe), halogens (Br, I) transition metals, TRU, uranium, lanthanides and actinide decay products in order to take them into account when an assessment of separation and waste management option should be made. For doing this, many aspects of their nature are studied, such as the long term radiotoxicity or their suitability for a certain separation process [40].

In the following graph can be observed the different radiotoxicity contribution from transition metal fission products from nuclear fuel based on UOX-51:

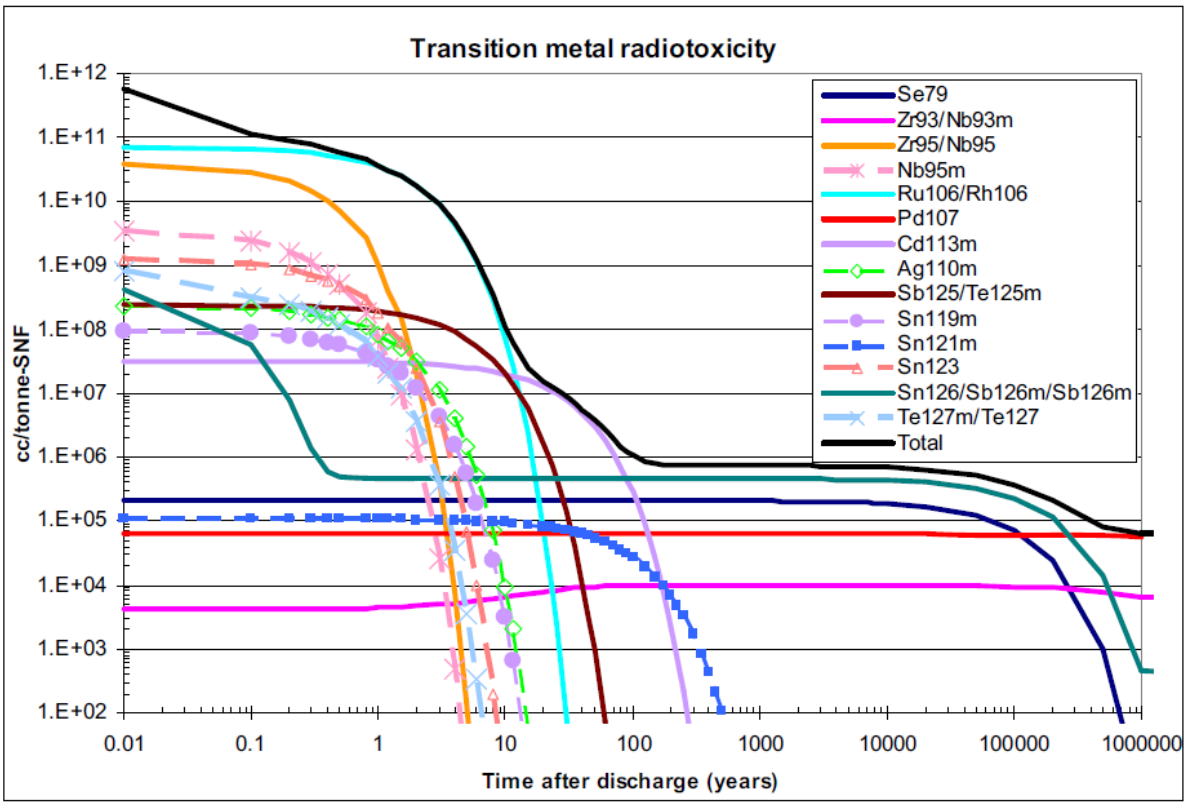


Figure 2-16: Radiotoxicity of transition metal FP from UOX-51 [40].

As can be seen, it seems reasonable to focus on the long term radiotoxic isotopes, i.e. those that continue to contribute to the waste radiotoxicity for more than 10 years after discharge (Se-79, Pd-107, Sn-126, …). This isotopes need to be considered in order to improve the design of final waste disposal (deep geological disposal, …) and to optimize the final disposal procedures (handling, reprocessing, …).

Also other aspects have been taken into account, such as gaseous effluents (krypton, tritium, radon, …) that have to be separated from solid wastes [40]–[42], precursors or daughter isotopes that share a common chain with a tracked isotope, burnup indicators (Nd-148), reactivity indicators,…

To summarize, a total of 73 isotopes have been selected to build the training model database, which are indicators of the main series and groups existing in nuclear waste (actinides, transition metals, lanthanides, …). The 73 chosen nuclides are the following[7]:

Table 2-7: Summary table of the 73 nuclides chosen as estimators to predict.

1	H3	**GP**	**16**	Sn126	**LL**	**31**	Sm151	**REA**	**46**	U232		**61**	Am243	
2	He4	**GP**	**17**	Sb125		**32**	Eu154	**GE**	**47**	U233		**62**	Cm242	
3	O16		**18**	Te125m		**33**	Eu155		**48**	U234		**63**	Cm243	
4	Se79	**LL**	**19**	I129	**GP LL**	**34**	Ho166m		**49**	U235	**REA**	**64**	Cm244	**DH NE**
5	Kr81	**GP**	**20**	Cs133	**BUC REA**	**35**	Pb206		**50**	U236	**REA**	**65**	Cm245	
6	Kr85	**GP**	**21**	Cs134	**DH GE**	**36**	Pb207		**51**	U238	**REA**	**66**	Cm246	
7	Sr90	**DH**	**22**	Cs135		**37**	Pb208		**52**	Np237		**67**	Cm247	
8	Y90	**DH**	**23**	Cs137	**DH GE BI**	**38**	Pb210		**53**	Pu238	**DH NE**	**68**	Cm248	
9	Zr93		**24**	Ba137m	**DH GE**	**39**	Bi209		**54**	Pu239	**DH BUC REA**	**69**	Cm250	
10	Nb93m		**25**	Ce144	**DH GE**	**40**	Ac227		**55**	Pu240	**BUC REA**	**70**	Bk249	
11	Tc99	**REA**	**26**	Nd148	**BI**	**41**	Th228		**56**	Pu241	**BUC REA**	**71**	Cf249	
12	Ru106	**DH GE**	**27**	Pm147	**REA**	**42**	Th229		**57**	Pu242		**72**	Cf250	
13	Rh106	**DH GE**	**28**	Sm146	**REA**	**43**	Th230		**58**	Pu244		**73**	Cf251	
14	Pd107	**LL**	**29**	Sm147	**REA**	**44**	Th232		**59**	Am241	**DH NE**			
15	Cd113m		**30**	Sm149	**REA**	**45**	Pa231		**60**	Am242m				

LL - Long Lived Fission Product

GP - Gaseous Product

GE - Gamma Emission Observable

DH - Decay Heat Observable

BUC - Burnup Credit Observable

REA - Reactivity Observable

BI - Burnup Indicator Observable

NE - Neutron Emission Observable

Assuming a smaller number of nuclides for the neural network design allows to enhance forecasting performance and, at the same time, allows to speed up both the database generation and training process. Subsequently, it also avoids the necessity of handling big data files, fact that can pose a problem in some memory-limited personal computers.

Finally, it is important to state that no information is lost in the database simplification, because in the previous table appears every isotope to be tracked for most fuel cycle analysis applications and spent nuclear fuel characterization.

[7] The list order is the actual order for the isotope columns of the database.

2.4. Database generation: burnup calculations in SERPENT2

A burnup model was generated to emulate the initial database that was used for preliminary research purposes, it consisted in the same geometry as the initial one had; however, several aspects were modified in order to obtain even more available data for the training process as well as to update the several nuclear data libraries existing in the model. SERPENT Monte Carlo code has been also used, but in a later version (2.1.32 vs 2.1.28).

In first place, it was necessary to build a copy of the Uppsala's model so differences between them could be studied (given that no direct access to the original SERPENT2 model was granted). Then, to achieve that, mostly three aspects of the model were available to tune, i.e. the fuel material vector, the libraries (decay, spontaneous fission yields, branching library, ACE library as a neutron cross section library and neutron fission yields) and the burnup parameters (so a higher resolution in the output could be obtained, i.e. more data available to train the neural network).

In this early model, uncertainties were explored to contrast the difference between nuclides inventories between both the Uppsala model and the one developed.

2.4.1. Uncertainties assessment

An experiment was designed to evaluate the fitting between both models. First a version of the input model was written (MOX and UOX were addressed separately as two different inputs for SERPENT2); secondly, the same calculation was launched 100 times so both average value and standard deviation could be extracted for each different nuclide. Then the database values were tested to fit within 2 sigma distance from the mean. Simultaneously, the repeatability and precision of the data was also tested (it was proved that no value had more than 1% of error when compared among the consecutive 100 outputs).

When the obtained standard deviation and mean were tested against the prototype database, it was also proven that employing a new set of nuclear data libraries was not an issue in terms of deviation, so newer versions of Evaluated Nuclear Data File (ENDF) were used, and the consideration of an isomeric branching data library was included (set bralib).

Table 2-8: Changes in the nuclear data libraries between the two models.

LIBRARIES	REFERENCE MODEL	NEW MODEL
ACE LIB	JEFF 3.1	ENDF/B-VII.1.
BRA LIB	NONE	
DECAY LIB	ENDF/B-VII.0	
NFY LIB	ENDF/B-VII.0	
SFY LIB	ENDF/B-VII.0	

JEFF-3.3 library was not used due to an anomalous behavior in the infinite multiplication factor calculations in the depletion problems when compared to other libraries (especially in the Pu-239 calculations), so ENDF/B-VIII.0 is used instead of JEFF for this update [43], [44].

The enrichment tested for uncertainty assessment were 3% of LEU for UOX fuel, and 4% content of Pu and Am for the MOX fuel:

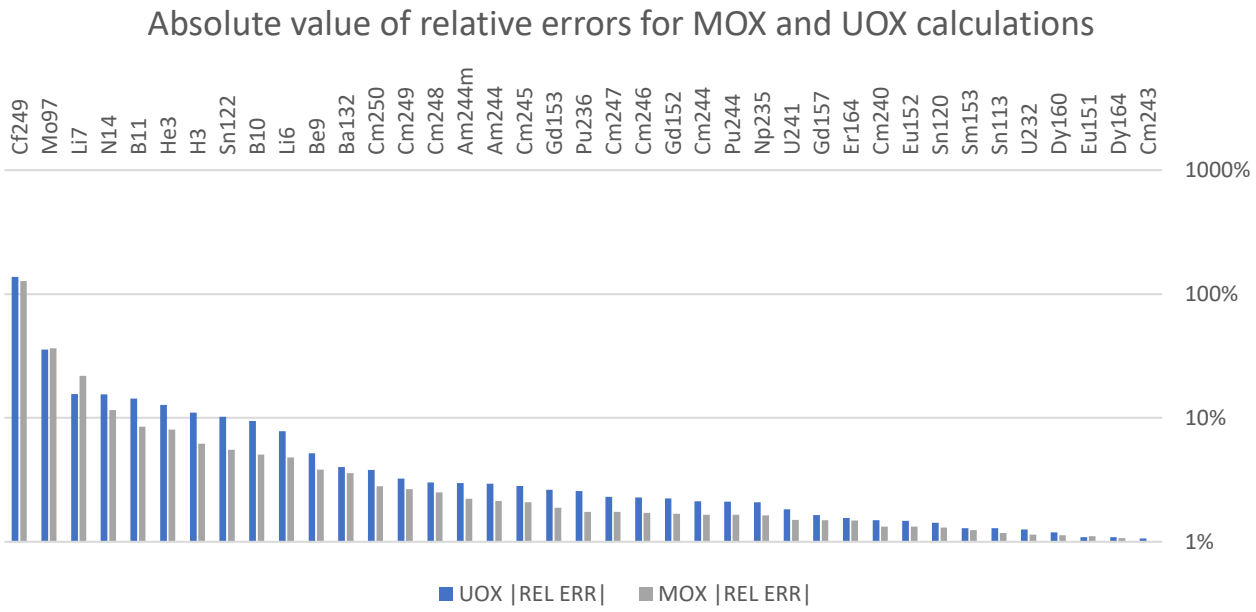


Figure 2-17: Absolute value of relative error for the different nuclides that match between both models and have more than 1% of error.

As it can be seen in Figure 2-17, most of the worst fitting nuclides are not important regarding the neural network design given that none of them are among the estimators referenced in the previous section. Others, like H-3, Pu-241, etc. are indeed selected to become future estimators for the final output of the neural network; nevertheless, it has been considered that they are in a tolerable uncertainty range for both fuel types. The following statements have been considered to be true:

1. The model geometry has been validated, so no more changes were implemented in this aspect.
2. The data quality and repeatability are good enough to start using this model as a database generator for the neural network.
3. Substitution of the older libraries does not suppose a detrimental action pertaining the nuclides uncertainties and implies an update to avoid deprecated nuclear data existing in older versions of them.
4. Fuel vectors for MOX and UOX are suitable for the design, so a Python script was made to calculate the different isotopic compositions for each fuel given as an input the initial enrichment for UOX and the americium and plutonium percentage over mass of heavy metals for MOX. The uranium and plutonium vectors remain the same that were used for the original Uppsala's database generation.

2.4.2. Database generation. Burnup parameters

Providing that a huge amount of data is needed for the neural network to train, the burnup cumulative steps have been augmented changing the step from 0.5 MWd/kgHM to 0.2 MWd/kgHM. To achieve this, the 30 days downtime every 10 MWd/kgHM from the prototype database was preserved. The downtime was modeled by setting the power to 0 and adding a decay period of 30 days, simulating refueling processes in PWR following the typical refueling pattern. No cooling time was added this time to any model.

Although the minimum burnup value considered for the database is 5 MWd/KgHM, the calculations were made from the very beginning of the cycle (BOC, with 0 burnup) for the sake of accuracy. However, whilst the addition of more steps to the burnup is possible and it will suppose an enhancement form the perspective of the data set resolution, a tradeoff exists with the computational time and the burnup steps. Given that changing the step from 0.5 to 0.2 resulted in raising the computation time from roughly 12.30 minutes for each enrichment percentage to 48 minutes, the value of 0.2 MWd/kgHM was settled to avoid excesive computational work.

Moreover, to take into account the spatial self-shielding effect in the burnable material in SERPENT2 (defined as a fuel and limited by the surface of a void that simulates the gap of the pin cell), a depletion zone division was implemented so the fuel pin was divided in 4 zones. This was done in order to consider that, even the if the fuel composition is homogeneous, the local neutron flux in the different lattice positions is not the same, so a different composition per each zone in the depleted fuel is expected [36].

2.4.3. Database generation. Fresh fuel isotopic composition calculation

The isotopic compositions for the different initially enriched fuel types have been also increased by changing the steps from 0.1% to 0.025% for UOX and from 0.2% to 0.025% for MOX fuel. This is expected to increase the resolution of the data sets and the prediction accuracy of the neural networks. Both minima and maxima remained the same (1.5% to 6% for UOX and 4.00% to 10.0% for MOX).

The UOX fuel has been simulated containing three nuclides: U-235, U-238 and O-16, so a fuel vector composition calculator was created to write the different weight percentages for every different enrichment.

The MOX fuel has been simulated containing the original fuel vector (Pu-238, Pu-239, Pu-240, Pu-241,Pu-242, U-234, U-235, U-238) plus the O-16 and Am-241.

The relative abundance of Am-241 was set in 1.25% relative to the total Pu existing in the fuel, following the indications of an IAEA report that stated that a typical range of 0.7-1.8 wt% is found in MOX fuel, so for the sake of simplicity the average of those two values was taken as a reference [45].

2.4.4. SERPENT2 input generation

The need of performing the calculations with several values of enrichment, and given the working principle of SERPENT2, implies that there should be the same amount of input files as grid points in the enrichment area exist. Taking into account that for MOX there are 241 enrichment steps (from 4% to 10% in steps of 0.025%) and for UOX there are 181 enrichment steps (from 1.5% to 6% in steps of 0.025%), 422 input files are needed to be run consecutively.

To achieve this, a Python script was coded, so that the different inputs could be written. This script basically consists of a sequential isotopic calculator which calculates the isotopic content for each fresh fuel pin, with the initial enrichment is modified by embedding the calculator in a loop that runs from the lowest enrichment to the highest per each fuel.

Then a logger function is called so the isotopic composition is written in a .txt file and then saved in the directory, later this same file is concatenated with another file containing other aspects of the model that are the same for every calculation (burnup, geometry, library directories, …). The final file is then renamed with a title containing the fuel type (MOX or UOX) and the enrichment with three decimals. Also, the name of the file is printed in the console along with other bash commands, to allow the continuous run of all of the files by using only one bash script in which is then copied the isotopic composition.

2.4.5. Computational aspects

Successive tests have been run in the SCK-CEN Newton calculation cluster. The dedicated nodes for running the calculations were 18 (CPU: XeonGold 6154 3.0 GHz, which provides 1476 non-hyperthreaded computing cores or 2952 hyperthreaded computing cores). In total, running all the calculations took up to 15 days. The different input files were launched separately in order to obtain different random seeds per each calculation, so no bias is expected to be present in the calculation.

2.4.6. Database arrangement

Database was then arranged by retrieving the different inventory data from all the SERPENT2 output files. The open source suite SerpentTools [46] was employed in the data parsing: first, “MDENS” block containing the final inventory mass density in g/cc was extracted from each dep.m file; then the data array was fit into a Pandas dataframe where the initial enrichment values were added. Finally, the whole dataset was split into a training and testing set with a 70/30 proportion by using a random distribution.

Given that no fresh fuel is expected to be calculated by the machine learner in ANICCA, a cutoff criterion was established, so only burnups values greater than 1 MWd/kgHM were included in the final datasets. This was done to lower the prediction error for low discharge burnup values, nevertheless, SERPENT2 calculation steps started from 0.2 MWd/kgHM to avoid accuracy issues.

2.4.7. Data Curation

In order to preserve a realistic approach to the whole burnup process in a PWR reactor, different decay steps were set so refueling stages could be taken into account. Nevertheless this is shown to cause misleading predictions in the neural network simulations for some isotopes, because there is no feature designed in the dataset that discriminates the decay time in this scenario.

The Samarium-149 is an important neutronic poison that cannot be removed from the dataset (and from the predictions). Given its absorption cross section, Sm-149 peaks at the end of every power downtime due to the decay of its father, Pm-149, and then, when the irradiation process is resumed its concentration begins to decrease because of the transmutation process. This causes the spikes that can be noticed in Figure 2-18:

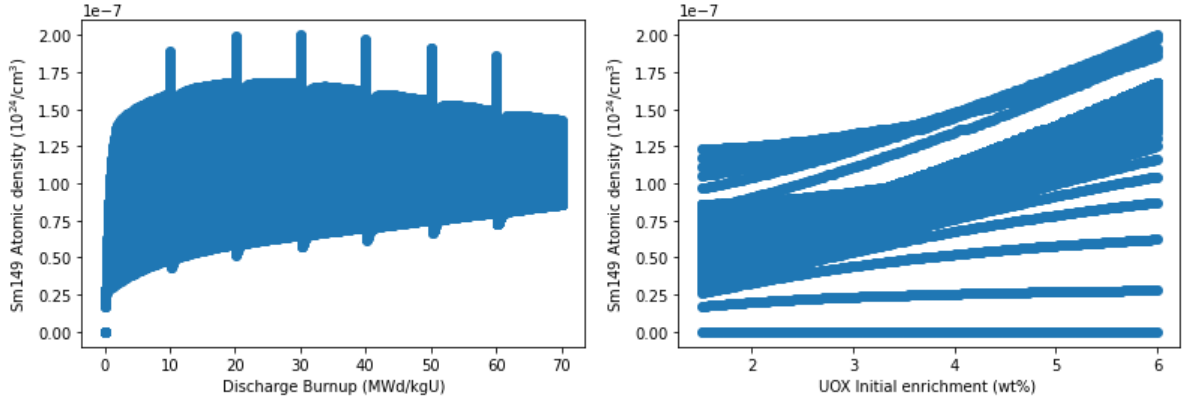


Figure 2-18: Atomic density for Sm 149 in UOX vs burnup and initial enrichment.

To smooth this heterogeneous data distribution a curation phase was conducted prior to the neural network training process. In this curation phase the Sm149 inventory values at the start of the irradiation steps were removed (those that match 10, 20, 30, 50, 50, and 60 MWd/kgU), the remaining gaps for that burnup values (NaN values) were then filled with different from zero value available in the Sm149 column, also median and average values were tested, but this method proved to be the best in terms of forecasting accuracy for this given neural network.

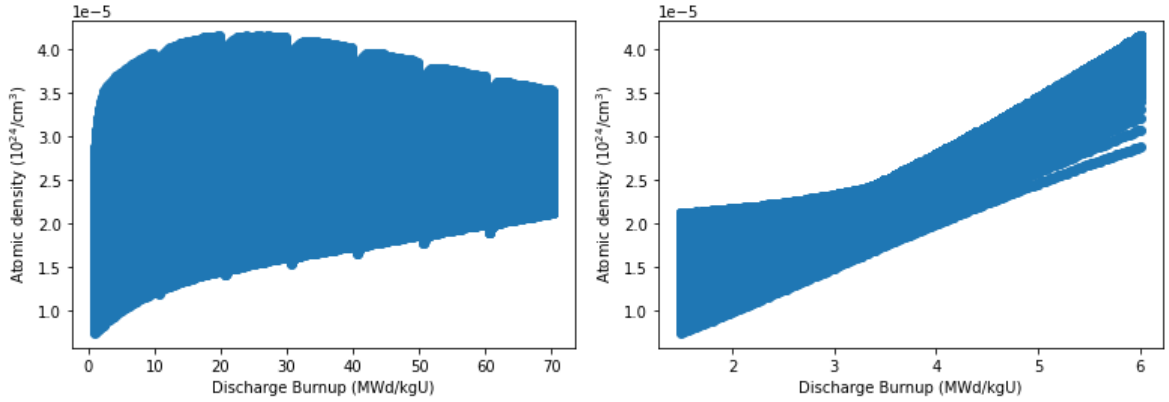


Figure 2-19: Sm 149 atomic density for UOX after the curation process.

2.5. Neural network model design

Both the UOX and MOX neural networks were built following an iterative process based on hyperparameter tuning: a Keras sequential model was studied in three different built-in optimizers, this sequential model was embedded in a function that continuously was modifying its architecture by changing layers and number of neurons per layer in order to provide the optimizers with different options. The optimizers then tuned the number of layers, number of neurons and learning rate for the most optimal configuration.

2.5.1. Hyperparameter tuning

The optimizers belonged to the Keras suite and were Bayesian Search, Grid Search and Hyperband algorithm. Grid search is able to search a given space of values for every hyperparameter to be tuned, the best combination is then chosen by taking into account the minimization of the loss function (which in this case was the mean squared error). Nevertheless, Grid Search did not deliver an optimal configuration given the limited range of search. Regarding Bayesian search bases, the next model upgrade in the previous results, whilst being a bit more prolific, the optimizer that reached the model with the lower error was Hyperband.

Hyperband optimizers bases the hyperparameter tuning procedure on the concept that the biggest loss in error is reached in earlier epochs of the model training, so Hyperband only conducts tests for a given configuration within a given range (min. and max. of layers, etc.) for a reduced number of epochs (implements early stopping techniques), when every possible configuration in that search space has been tested, more training epochs are then added and the algorithm keeps looking for its candidate between those models that were more prolific in the previous steps with fewer training epochs.

Also references [47], [48] confirmed empirically the better performance of both Bayesian and Hyperband searches when tested against classical grid or random search algorithms.

The previous information to model the input of the Hyperband tuner was the optimizer type, the activation functions and the kernel initializers. Also, decisions such as the inclusion of dropout layers were to be made.

Adam was chosen to be the optimizer required to train the model, it is an alternative for the classical stochastic gradient descent method that is able to change the learning rate (something that stochastic gradient does not do), this method uses different adaptive learning rates for different parameters of the neural network and it has been empirically recognized as one of the most effective optimizers currently available. Besides, less memory is needed and a smaller computational burden is expected by using this algorithm [49],[50].

The selected activation function for the hidden layers was the rectified linear unit or ReLU, the choice was based on the simplicity concept: ReLU is the most simple nonlinear activation function in the required activation span (from 0 to 1) that is recommended for regression problems (indeed, it is the most used in every application). As a matter of fact, the bibliography also recognizes ReLU as the activation function with the best performance for regression tasks [51]. Last activation function (for the output layer) was chosen to be linear; this was done in order to avoid negative numbers in the predictions.

The kernel initializers are the ones providing initial values for the weights in the different neurons. They were chosen to be of the “He normal” kind, which is proved to work well along ReLU activations functions for these specific datasets in several trial-and-error runs.

Dropout layers are intended to regularize a deep neural network thus avoiding overfitting by adding statistical noise throughout the training process by deliberately ignoring random layer outputs (certain units are temporarily shut down). However this option was not taken, given that no evidence of error decrease was found for regression problems with dropout layers [52].

Hyperband tuner then ranked the 10 best estimate models with their corresponding number of layers and neurons plus the recommended learning rate. Then the number of epochs and the batch size were set manually by observing training and validation losses evolution vs epochs.

2.5.2. Data scaling

Given the data type, it was decided to normalize all the available data in a scale from 0 to 1 that avoids causing the ReLU activation functions to saturate. This method is unsuitable in presence of outliers given that the resolution of the scaled data is based on the minimum and maximum values, nevertheless, these datasets had no outliers given that no noise or experimental error could be recollected because it was obtained by computational simulations. Scikit Learn MinMaxScaler was used for this task and it was applied to both testing and training sets. MinMaxScaler has the capability of scaling the whole dataframe per feature (which dimensionally matches a column of the dataframe), so only four scalers were used (one for targets and other for features for both testing and training sets).

2.5.3. UOX Neural network

The Hyperband tuner obtained the next optimal configuration for the UOX dataset:

Table 2-9: Hyperparameter tuner for UOX neural network.

Number of hidden Layers	6
Num. of neurons in Layer 0	44
Num. of neurons in Layer 1	140
Num. of neurons in Layer 2	236
Num. of neurons in Layer 3	236
Num. of neurons in Layer 4	80
Num. of neurons in Layer 5	176
Learning Rate	0.001
Score (Mean Squared Error)	1.8251781511935405e-05

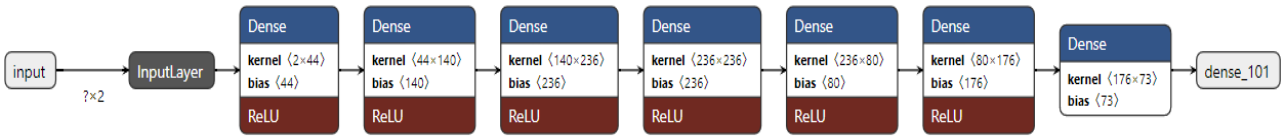


Figure 2-20: Architecture of the neural network model for UOX. Note that the output layers have the random name of "dense_101".

Then, the number of epochs was tuned during the training session into 100 epochs, which also matches the maximum number of epochs for Hyperband, with a batch size set to be 64. A batch size of 128 was chosen for evaluation on testing data.

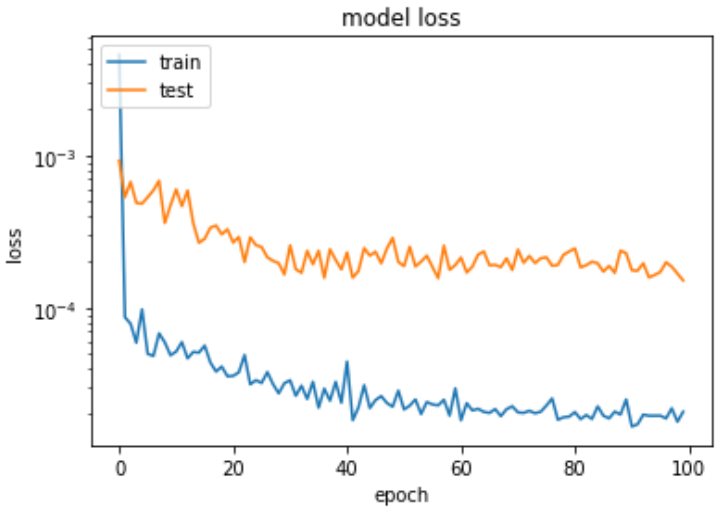


Figure 2-21: Mean Squared Error vs training epochs for UOX neural network.

Note in the Figure 2-21 that the validation error is higher than training losses which is normal. It cannot be considered an aggressive overfitting problem given that no increase of the validation loss is appreciated in the latest training epochs.

Table 2-10: Loss results in UOX neural network.

Last validation loss (validation split of 0.2)	**1.5131e-04**
Last training loss	2.0790e-05
Error in validation on testing set	4.5416e-05

As can be noticed, the final training loss is similar to the value predicted by the Hyperband (1.82517e-05). Slight discrepancies are always expected to happen given the stochastic nature of these processes. The difference between the error in the testing set and the error in the training set can also be noticed, however, this was expected since the test loss estimates the model performance on unseen data, which is always going to be worse.

2.5.4. MOX Neural network

The Hyperband tuner obtained the next optimal configuration for the MOX dataset:

Table 2-11: Hyperparameter tuner for MOX neural network.

Number of hidden Layers	5
Num. of neurons in Layer 0	56
Num. of neurons in Layer 1	32
Num. of neurons in Layer 2	104
Num. of neurons in Layer 3	284
Num. of neurons in Layer 4	32
Learning Rate	0.0001
Score (Mean Squared Error)	2.0266283172531985-05

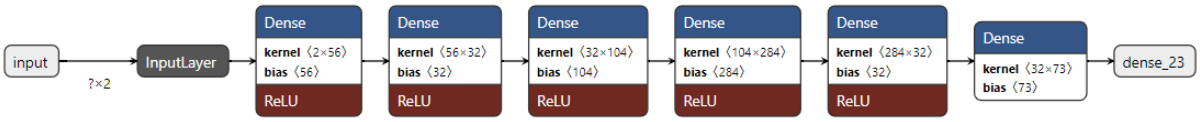


Figure 2-22: Architecture of the neural network model for MOX. Note that the output layers have the random name of "dense_23".

Note that in this case, the learning rate was ten times smaller than for the UOX neural network. Following the steps of the previous neural network, the number of epochs was also tuned during training session into 100 epochs which also matches the maximum number of epochs for Hyperband, with the batch size was set to be 64. A batch size of 128 was chosen for evaluation on testing data.

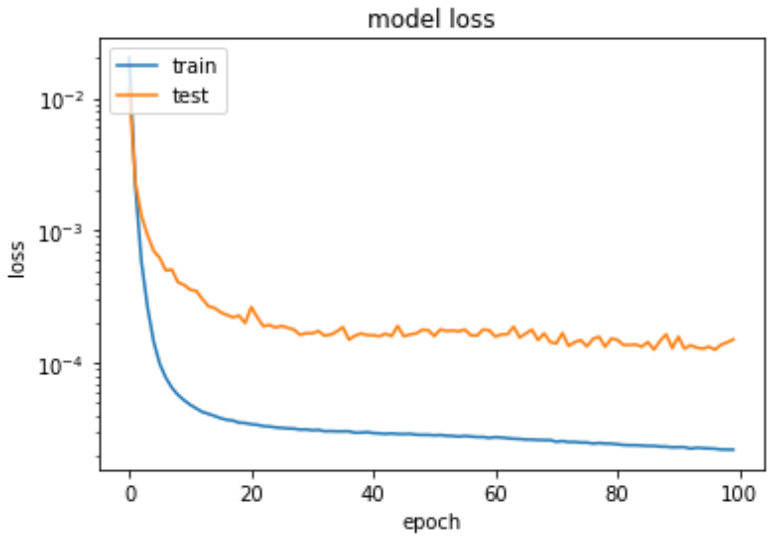


Figure 2-23: Mean Squared Error vs training epochs for MOX neural network

Once again and according to Figure 2-23, it cannot be an aggressive problem of overfitting for the MOX neural network. Nonetheless, this time the score is a little bit lower in comparison with UOX neural network. Note that now that the learning rate is 10 times slower and the loss function has been smoothed in comparison with the UOX neural network.

Table 2-12: Loss results in MOX neural network.

Last validation loss (validation split of 0.2)	**1.5069e-04**
Last training loss	2.2166e-05
Error in validation on testing set	5.7768-05

As can be noticed, the final loss is similar to the value predicted by the Hyperband (2.0266e-05).Once again, slight discrepancies are always expected to happen given the stochastic nature of these processes. At first sight, it seems that the neural network for MOX has poorer performance than the one for UOX; even so, this was expected given the complexity of the inventory for the second one. However, this aspect of the neural networks will be in the following.

2.6. Model integration in ANICCA code

Since the ANICCA code was written in Python, the code modularity is delivered by the object-oriented programming, so this advantage was used for implementing the machine learner approach within the original code.

To maintain the original functions in the previous version, the new functionality was added by creating new class variables in the original blocks, so no original functions had to be modified. Besides, the machine learner was added in a flow control structure to properly separate the two different inventory calculation options, i.e., the previous version (CRAM), and the new one (ML).

To provide an overall view of the modifications that were performed, a scheme is provided in Appendix A.

2.6.1. Input file modifications

ANICCA is capable of properly reading the scenario and components initialization data from a human-readable data-serialization language file like YAML or JSON, so in order to create the new necessary inputs, two tags were added in the scenario and power plants files. The first one is declared in the component files for the different power plants: the tag “ML”, which is initialized as the string containing the name of the neural network serialized model file path.

The second one is declared then in the scenario initialization file, specifically in the nuclide data input structure. This second tag is a string containing the name of the list for the new 73 specified isotopes. This was done because as previously stated, the neural network only has a prediction capability for the isotopes that were contained in the training database.

Besides and pertaining to future benchmarking tests, it seems fair to provide the CRAM functionality with the same isotopes for the calculations that are in the ML model, so an even comparison between the two models’ performance could be performed. This string was set to be “*Casas_iso_ml*”.

2.6.2. Isotopes list integration

The 73 studied isotopes were inserted in the scripts of ANICCA at the same level and places where the existing isotopes lists were, so then the code could work only on these nuclides. To achieve that, the ZZAAAm isotope name convention was adopted, where Z is the atomic number and A the mass number for a given isotope. Also, a new entry for the isotope dictionary[8] was assigned for this list.

2.6.3. ML input acquisition

ANICCA fuel cycle code traditionally extracts the burnup and enrichment from the different existing libraries in its files, so two new variables were needed: the total burnup for the given scenario specifications, and the initial enrichment from the designated fuel. These two variables could be easily extracted from the nuclide data unit, which previously collected them from the aforementioned libraries.

Given that the enrichment is directly provided by the libraries in fractions, no modifications or scaling were needed, however, the burnup is traditionally provided to the irradiation module in burnup steps, so a new cumulative variable was added to collect and summarize the total burnup steps before passing that value to the new irradiation module.

Both inputs were kept in the form of class variables for the nuclide data module, so they could be then directly invoked from the required modules (i.e., the irradiation module).

[8] ANICCA uses the same vector for making the different irradiation calculations. The vector is previously formatted via an isotope dictionary depending on the required observables by the user.

2.6.4. ML module embedding and output configuration

Keras trained models can be saved into hierarchical data format .h5 files, so they can be later employed in other code snippets to make predictions without the necessity of training them again, which saves computational time and resources.

Apart from saving the models, also the data scaling information must be saved so calculations in the same normalization range are preserved. The scaling data information (mean and standard deviation) is saved in the pickle format, which was conceived for serializing Python objects. This was the chosen option given that other alternatives such as importing the Scikit Learn preprocessing package into the ANICCA code would bring compatibility and portability problems, plus the fact of increasing the non-productive computational time demand (loading and processing external modules).

The serialized files can then be saved into the ANICCA data directory so they could be accessed from the irradiation module, where the ML calculation function has been implemented and wrapped in a control structure that prevents the simultaneous activation of both the ML and the CRAM function and prevents the ML from performing fresh fuel calculations (with a burnup input of cero), which is not desired for this task. Besides, an alternative was added to properly distinguish between MOX and UOX fuels, so that the proper ML model could be reached.

The output then is converted from mass density (neural network working units) to weight fractions (ANICCA working units) and passed to the isotope vector, which, if done correctly, is at the time of the calculation run loaded with the 73 desired tracking isotopes. The rest of the program flow remains intact given that no other functionality is needed, and the former capabilities have been preserved.

For demonstration purposes regarding the MOX model, the fuel vector which was used for generating the training database has to be fixed as the vector for MOX in ANICCA, i.e., the initial isotopic composition of the MOX fuel. This was done in order to properly assess the impact of the CRAM substitution when studying a closed fuel cycle, which requires predicting capabilities for MOX isotopic composition. Actually, in the ANICCA environment fuel vectors for MOX can be modified allowing different reprocessing compositions as well as different Am-241 contents (which decays from Pu-241 and builds up during the storage or cooling times).

On the other hand, the ML model cannot reproduce every initial composition for MOX fuel at this current level of development, given that the databases only provided training data for a specific plutonium vector. Although predicting every vector would be possible, it would require another neural network model with more features as inputs, such as the different plutonium isotopes that make up the fuel vector. Nonetheless, future capabilities for an enhanced ML model will be discussed in the conclusion chapter.

2.7. Benchmarking design

In order to assess the performance for the deployed ML models, two input scenarios for ANICCA are required, so that the differences between the previous CRAM and the new ML approach could be detected. The first scenario, which was designed to test the UOX ML model, simulates an open fuel cycle; the second one, devoted to the MOX ML model testing, simulates a closed fuel cycle.

The first shared step for both scenarios was to update the different fuel libraries via ALEPH2 [16], so that they were based on the same nuclear data library (ENDF/B-VIII.0) as the SERPENT2 datasets to avoid potential bias from nuclear data.

2.7.1. UOX test scenario

For the UOX test, a simplified scenario was designed, this open cycle scenario involved two PWR reactors, both with the same following characteristics:

Table 2-13: Characteristics of the UOX test scenario.

Electric power	**0.72 GW**
Thermal power	2.2 GW
Reactor core mass	52.8 t
Burnup	60.1 MWd/kgU
Load Factor	0.9
Residence Time for SNF	5 years
Starting in	2025
Time steps	400 x 1 year

The main objective is to observe the error propagation along the calculation years, so that comparisons based on a time evolving scenario can be established and differences between CRAM and ML approaches performance can be detected.

2.7.2. MOX test scenario

For the purpose of testing the MOX model, a scenario taken from reference [13] was modeled. In this case, a whole closed cycle scenario, based on the phase out of the Belgian nuclear fleet, is simulated.

In this scenario, no new nuclear plants are built and the lifespan of the reactors are defined in the following table:

Table 2-14: Reactors' characteristics [13].

NPP	Operation time		Power (MWe)	Power (MWth)
Doel 1	1975	2025	433	1311
Doel 2	1975	2025	433	1311
Doel 3	1982	2022	1006	3054
Doel 4	1985	2025	1038	2988
Tihange 1	1975	2025	962	2873
Tihange 2	1982	2023	1008	3064
Tihange 3	1985	2025	1038	3000
EFIT	2050	2136	154	400

MOX is being used in the Doel 3 and Tihange 2 units with the following fuel specifications:

Table 2-15: Fuel specifications [13].

NPP	Average Burnup (MWd/kgHM)	Fuel type
Doel 1	30/45	UOX
Doel 2	30/45	UOX
Doel 3	45	UOX/MOX
Doel 4	45	UOX
Tihange 1	30/45	UOX
Tihange 2	45	UOX/MOX
Tihange 3	45	UOX
EFIT	130	Pu + Minor Actinides

EFIT an ADS (Accelerator Driven System) whose main purpose is to burn the generated MA during the lifespan of the nuclear fleet. EFIT is an advanced reactor and has been considered out of scope, so it has not been included in the MOX benchmark. The flow chart for the scenario simulated in ANICCA is shown in Figure 2-24.

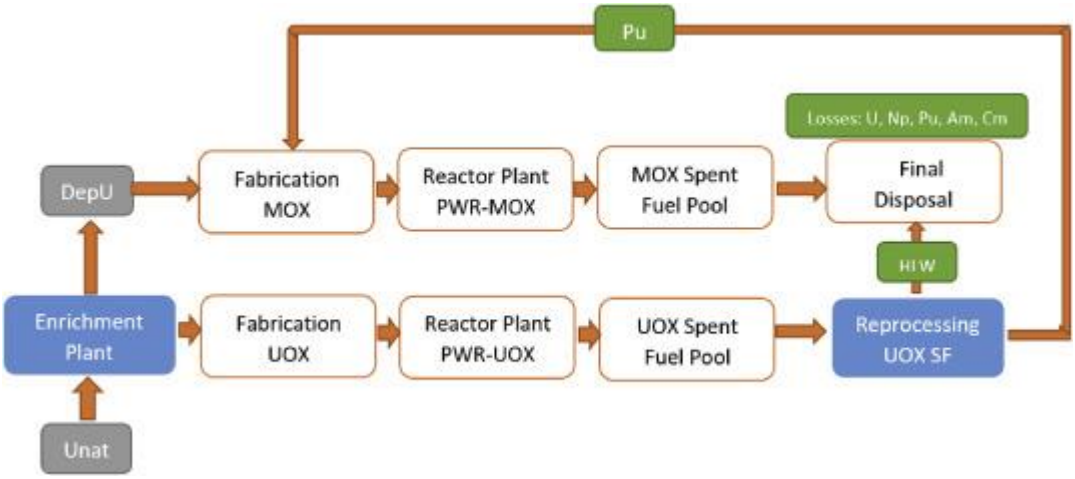


Figure 2-24: Scenario flow chart in ANICCA [13].

In the closed cycle simulation, both UOX and MOX models have to work overlapping each other, so it is an ideal condition to test the correct integration of both models in ANICCA's original flow.

3. RESULTS

In this chapter both the results of the neural network in terms of prediction error and the benchmarking between ANICCA's original irradiation method and the new implementation of a ML model are shown and described.

3.1. Neural networks performance

The performance of both neural network models has been quantified in terms of error for both the overall prediction error for every isotope and for each burnup and enrichment grid points. For the sake of convenience, only the most significant isotopes are presented here. In the following color maps, the error distribution in percentage per each grid point for six representative studied nuclides (which are important as heat generation indicators, burnup indicators, or gamma/neutron emission indicators) can be seen. Please note that, whilst the error shall not be sketched as a negative value, this modification over the error definition allows to tell whether the density values for that given isotope is going to be overestimated and underestimated by attending the error's direction.

The errors of the UOX final inventory prediction seem to be close to zero in most cases and for the vast majority of burnup and enrichment values, yet a significant increase in the error value for Pu-238 and Am-241 can be noticed, especially at high enrichment and low burnup values.

At the same time, it can be stated that in some nuclides such as Pu-239 an error increase is found at burnup values where the decay steps were set in SERPENT2. In the rest of cases and as a general remark, it can be stated that error is prone to increase mostly at high enrichment-high burnup values and also at low burnup values for both very low and very highly enriched fuels (referring to the existing enrichment scale).

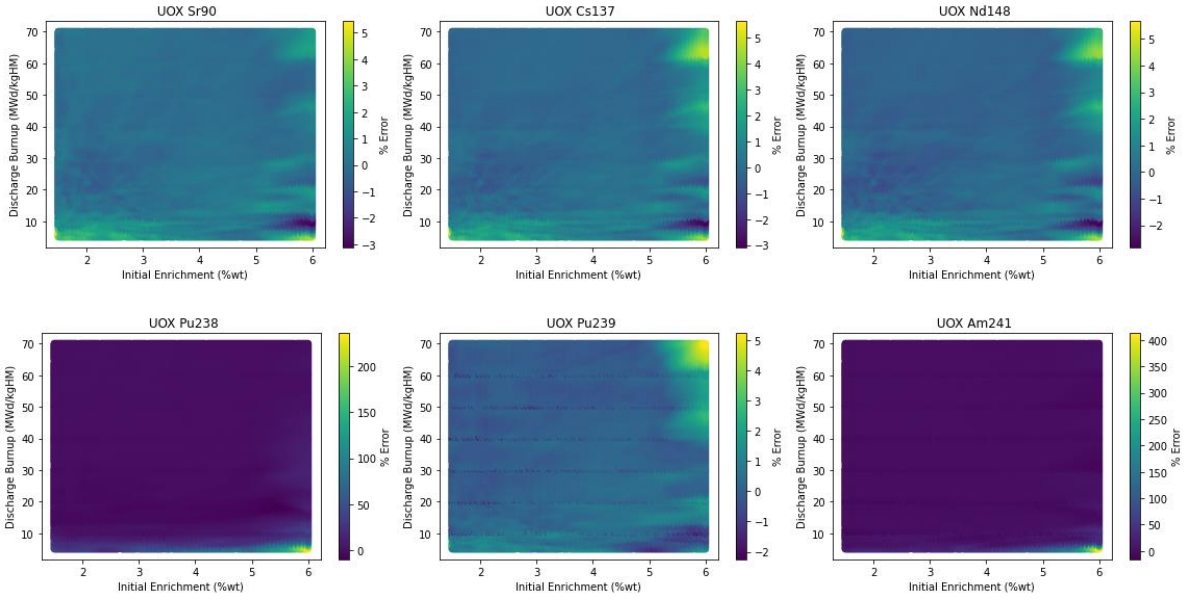


Figure 3-1: Error (%) for six representative tracked isotopes in the UOX N.N. model

The error for the same nuclides for the MOX fuel type are shown below:

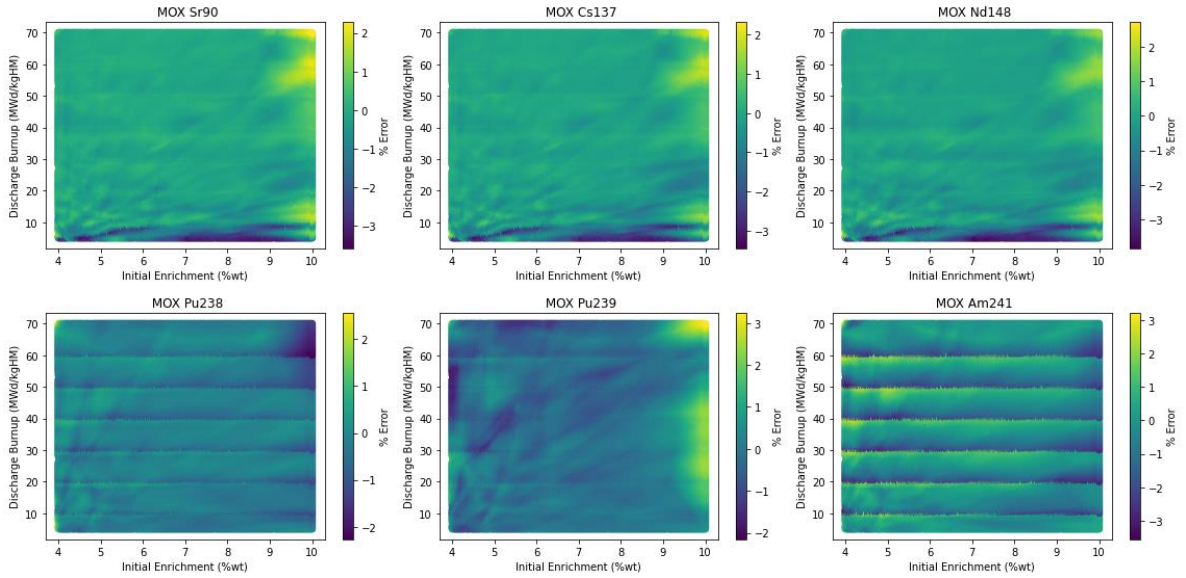


Figure 3-2: Error (%) for six representative tracked isotopes in the MOX N.N. model

In this case, similar patterns can be observed for Sr-90, Cs-137 and Nd-148. On the other hand, the other three nuclides show a different pattern: Pu-238 and Am-241 have a similar error value but also have spikes for the burnup values in which there is a decay time; Pu-239 experiments an error increment for fuel with high Pu and Am content for every studied burnup value.

A general underestimating trend exists for the first three isotopes for low burnup values that did not appear in the UOX model. It shall be noticed that wherever a decay step exists, the predicted quantity is firstly overestimated and then, when the decay step ceases, the quantity tends to be underestimated.

As a general remark for both types of fuel, it can be stated that the lowest error values can be found within intermediate values for both the initial enrichment and the discharge burnup, with the exception being the burnup points with a decay step .

It should also be noticed that there seems to exist a trend for the average error that causes it to increase as the atomic weight of the nuclides increases.

Then, to study the evolution of the error for a given burnup and enrichment value, a scatter plot of atomic mass vs number of protons can be found in Figure 3-3 for three reference values of enrichment and three reference values of burnup for both UOX and MOX. These values were selected to provide an overall view of the enrichment span of the N.N. in typical values of burnup for spent fuel characterization and fuel cycle analysis, i.e., real or close to real discharge burnup values.

This way, a feedback of the neural network performance based on real average enrichment and burnup values that can be found for research and commercial reactors is provided. Besides, a comparison can be established with previous work found in state-of-the-art references.

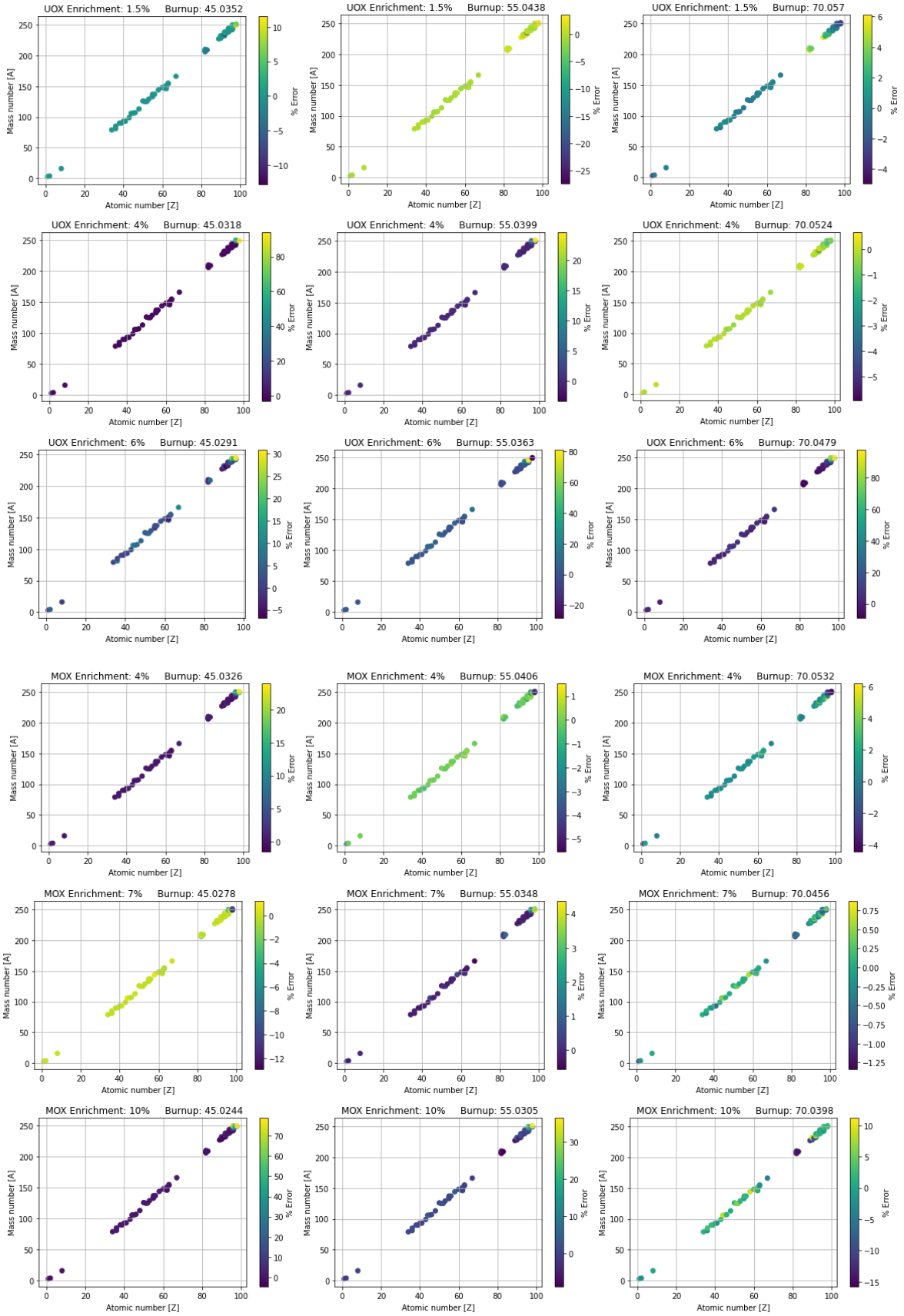


Figure 3-3: Relative error (%) for low, medium and high enrichment values for UOX and MOX in discharge burnups values within range for spent fuel cycle analysis. Burnup units are in MWd/kgHM.

As can be seen, the error on average is kept in values close to zero for almost every isotope plotted for every different burnup, enrichment, and fuel type. The worst estimate can be found for highest enrichment values. Conversely, it can be noticed how the nuclides with greater mass are prone to be deviated from the real values.

Indeed, it is important to notice that for the sake of convenience and to avoid visualization problems, nuclides with an error greater than 100% have been given the consideration of outliers and been discarded of the previous plots. Table 3-1 provides the relation of nuclides discarded following the same order that the plots have.

Table 3-1: Discarded outliers with an error greater than 100% in their respective positions.

UOX (IE (%), BU (MWd/kgHM)) -Isotopes-		
(1.5,45)	(1.5,55)	(1.5,70) U235
(4,45) Cf250,Cf251	(4,55)	(4,70)
(6,45) Cm246,Cm247,Cm248,Cm250, Bk249,Cf249,Cf250,Cf251	(6,55) Cm247,Cm248,Cm250,Bk249, Cf251	(6,70) Cf251
MOX (IE (%), BU (MWd/kgHM)) -Isotopes-		
(4,45)	(4,55)	(4,70)
(7,45)	(7,55)	(7,70)
(10,45) Cf251	(10,55)	(10,70)

Greater deviations for isotopes with an atomic mass greater than 246 are expected, with the exception of U-235 for high burnup rates at very low enriched uranium in UOX fuel and with the MOX N.N. showing a smaller average error for every grid point than UOX. Nonetheless, the error increase is localized for these isotopes for certain BU and IE values, which are in bounded together.

It should be considered as well that all suppressed outliers were overestimated. Indeed, very few isotopes are underestimated, and this trend is only noticeable for MOX fuel. Pb-206, Pb-207, Pb-208, Bi-209, Ac-227 and Th-229 have a negative deviation of more than 10% but no greater than 15% only for points with IE greater than 9% and BU greater than 50 MWd/kgHM.

Given that the 9 existing outliers are repeated in different grid points for the UOX fuel, the evolution of the absolute error for these nuclides with the burnup has been independently tracked, with the objective of assessing the valid utilization range for the neural network.

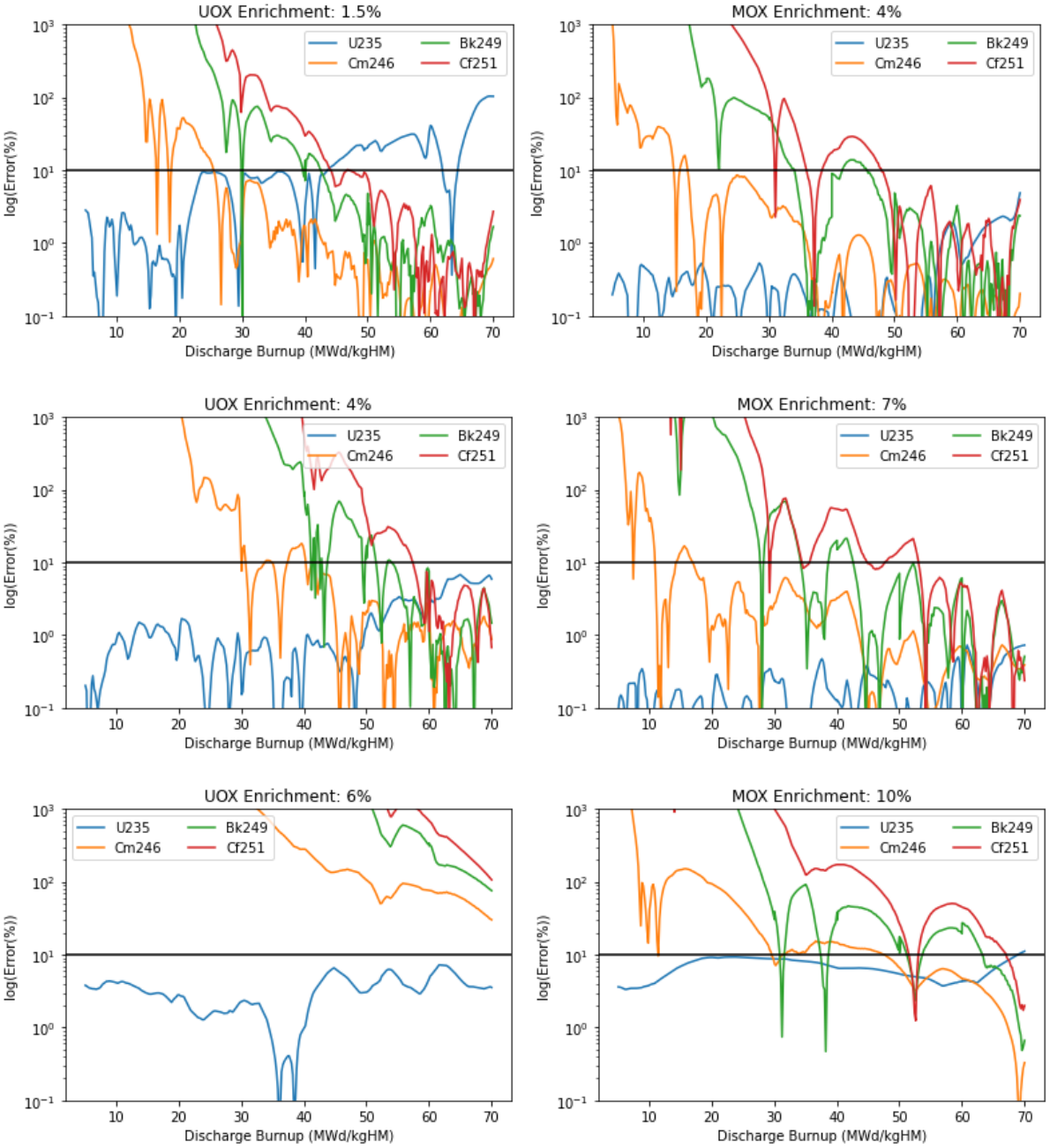


Figure 3-4: Relative error (%) for the outliers, being Cm246 the one with the lowest error values and Cf-251 the one with the largest, every other isotope of its respective element (Cm and Cf) falls in between both.

As can be stated in Figure 3-4, U-235 error only exceeds 10% (black line) for 1.5% UOX enrichment with burnup values greater than 40 MWd/kgHM. It can be seen how the error for the other outliers becomes manageable for low to medium enrichments for both UOX and MOX fuels with high discharge burnup values, i.e., greater than 50 MWd/kgHM approximately.

Besides, the error for the heavier outliers seems to increase with the enrichment and to decrease with the burnup for both fuel types.

To summarize the remaining information about the models, a statistical overview is provided in Table 3-2.

Table 3-2: Highest relative error (%) for the 80% of the population for different burnup ranges at regular enrichment values.

	UOX (3-5%)			MOX (5-9%)		
Burnup (MWd/kgHM)	25-38	38.1-50	50.1-70	25-38	38.1-50	50.1-70
Percentile 80 (%)	9.86736	7.08457	5.40507	9.25407	6.9178	5.72159

For the most habitual enrichment values, it can be stated that the 80 % of the predicted isotopes for values that fall in between typical discharge burnups are below 10% of relative error. Also, the worst prediction is for Cf-251 in the three burnup ranges for both fuel types. For more information, Appendix B contains the list of percentiles for every tracked isotope.

3.2. Benchmark with ANICCA

In this section, the different results of ANICCA's ML version are presented. The performance of the code over an open cycle scenario is assessed via the comparison of the evolution of the total inventory for the total uranium, total plutonium, transuranic elements (TRU), minor actinides (MA) and fission products (FP)[9] segments.

Is important to remark again that the open cycle scenario was developed to test only the UOX ML model's performance.

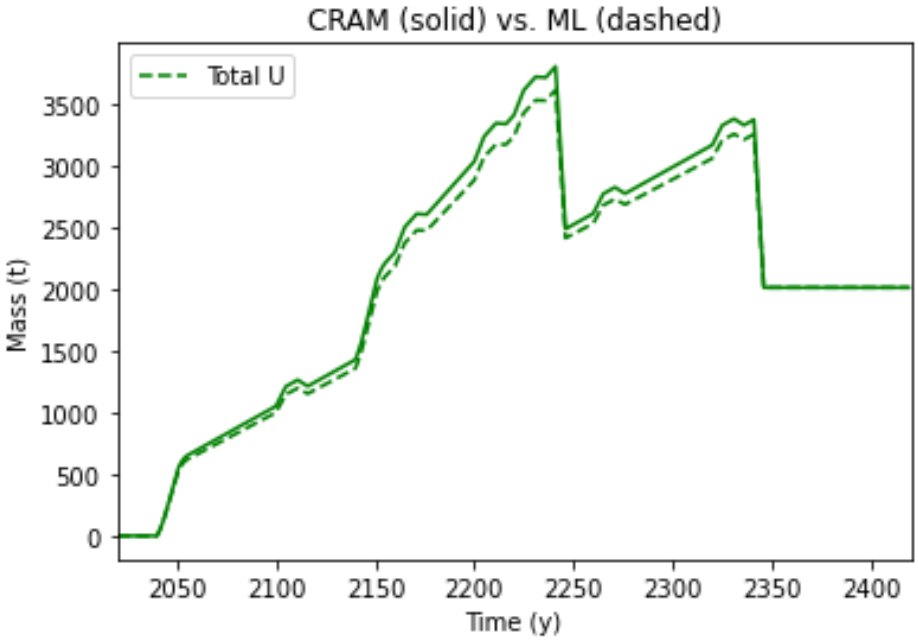


Figure 3-5: Evolution of the inventory of total Uranium for the open cycle scenario.

As can be seen in Figure 3-5, deviation over the previous CRAM-based model is almost none, and in some periods, mainly when the total amount increases, this deviation is slightly increased too. Nonetheless, the deviation is not building up during the simulation, so no error propagation issues can be detected. This same observation can be applied on the other groups shown in the Figure 3-6.

Please, note that the time periods when the mass is constant match the periods when the simulation is close to finish (time target reached) or no power plants are in operation (such as the beginning of the simulation).

[9] From this comparison, fission products with an atomic number smaller than or equal to 8 have been removed from the group, given that ANICCA by default is solving CRAM based only on Heavy Metal values (so no nuclides such as oxygen are considered).

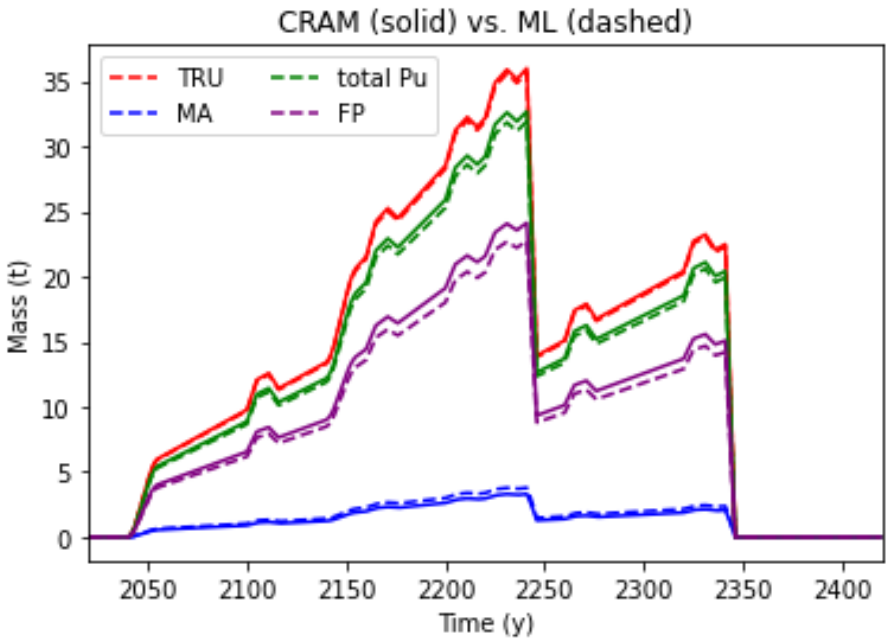


Figure 3-6: Evolution of the inventory of TRU, total PU, MA, and FP over the simulation time for the open cycle scenario.

In Figure 3-6 the minor actinide mass is slightly overestimated, while for the rest of the groups the mass is underestimated. As in the case of the uranium total mass, the deviation is more noticeable in the build-up periods, while when the nuclides are removed from the scenario these mismatches are almost zero.

Now, if the relative deviation of the results is plotted (Figure 3-7), it can be noticed how the error is kept around 5% for every studied group except for the minor actinides. Nonetheless, the MA group does not have a very big amount of mass when compared to the rest of the groups, so it is clear that relative deviation is prone to increase notably. Another relevant aspect to extract from this plot is the fact that deviations are not increasing with the simulation time.

However, if the deviation is put in tons, nothing relevant can be found from the MA deviation as it is evolving as the other groups, so the higher values for the relative deviation can be explained with the relatively small total mass for this group.

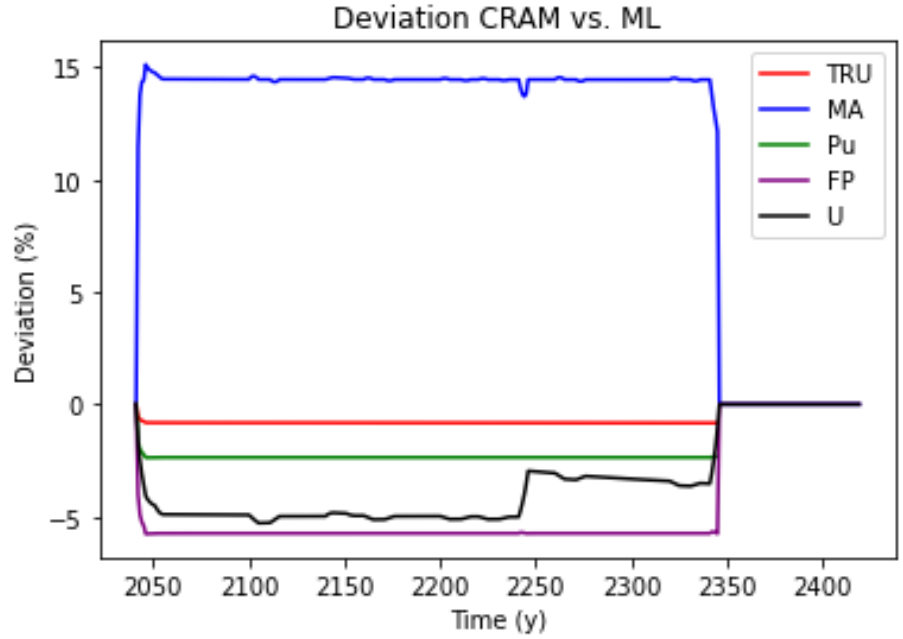


Figure 3-7: Relative deviation of the open cycle scenario groups.

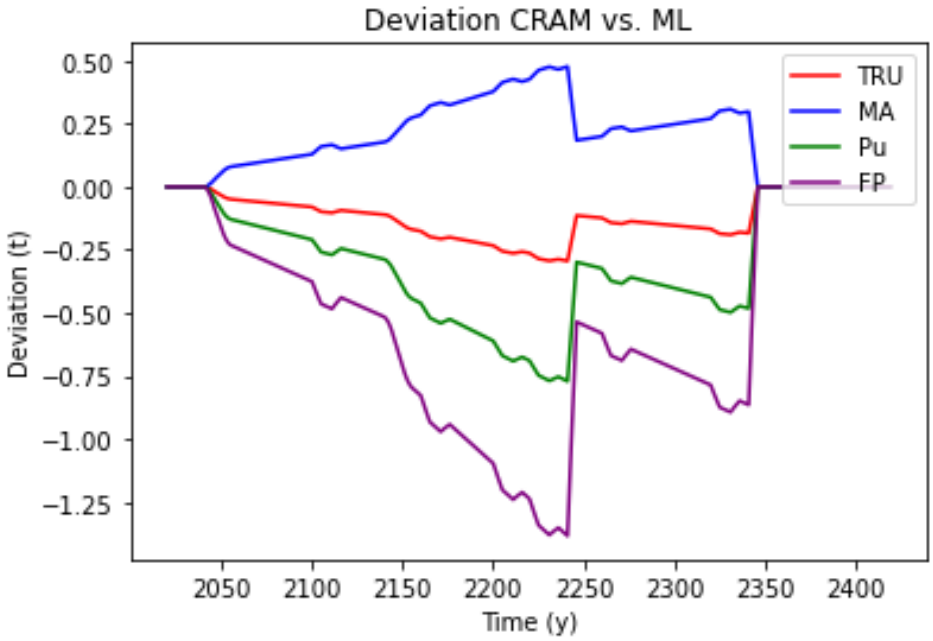


Figure 3-8: Absolute deviation of the open cycle scenario groups. Note that U has been removed from this comparison due to its large amount.

The same figures are now shown for the closed cycle scenario. In this scenario both UOX and MOX models are implemented in ANICCA, so a fair comparison between the previous CRAM-based model and the newest version can be established.

In Figure 3-9, the total uranium mass evolution during the closed cycle scenario can be observed. Once again, despite the different evolution rates and quantities reached, the ML is located below the CRAM marks. Also the deviation seems to be constant in periods where the total amount is not changing significantly but tends to be reduced when strong incremental or decrement rates are given.

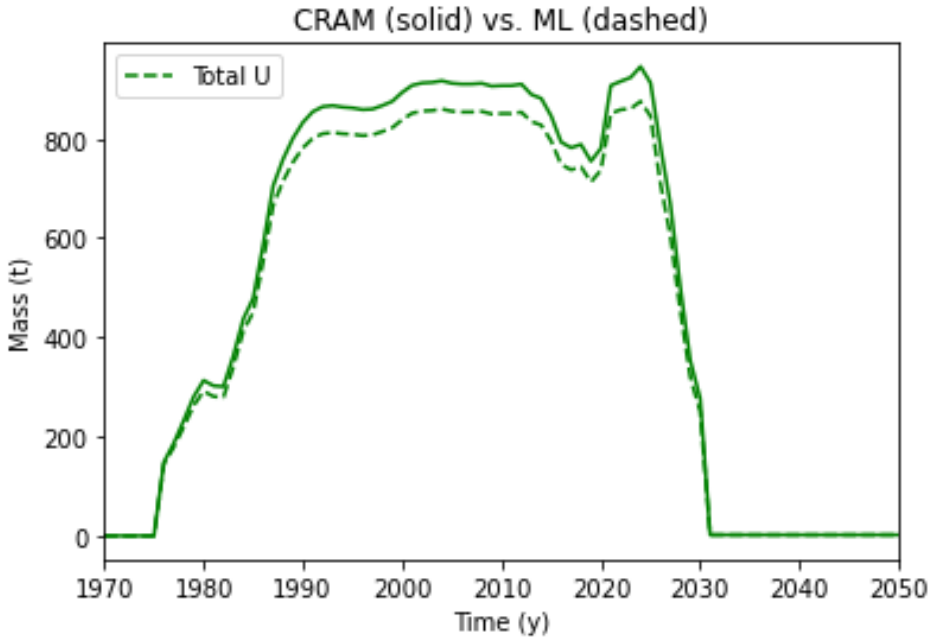


Figure 3-9: Evolution of the inventory of total Uranium for the closed cycle scenario.

For the rest of the studied groups, the open cycle scenario pattern is noticed again: every total mass amount is slightly underestimated except for the minor actinides, which are a little smaller for the ML approximation.

In the relative deviation, a greater value is perceived in general for every group, so unlike the case of the open cycle scenario, bigger mismatches were found, the relatively high deviation for the MA draws the attention, (over 40% at the beginning of the simulation). Once again, this is due to the small total mass value for said group. At the same time, in this case FP and MA deviations are not constant as in the case of the open cycle scenario.

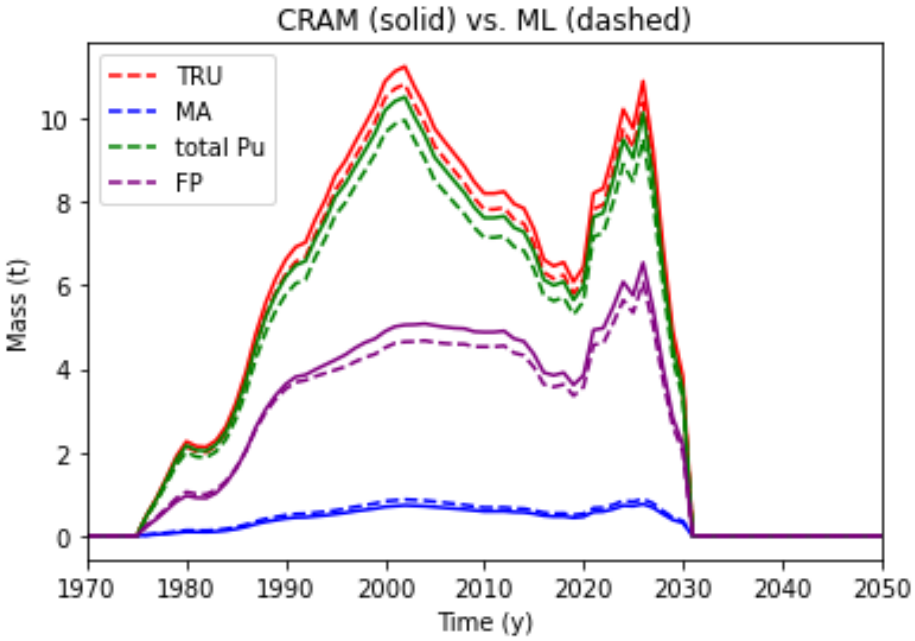


Figure 3-10: Evolution of the inventory of TRU, total PU, MA, and FP over the simulation time for the closed cycle scenario.

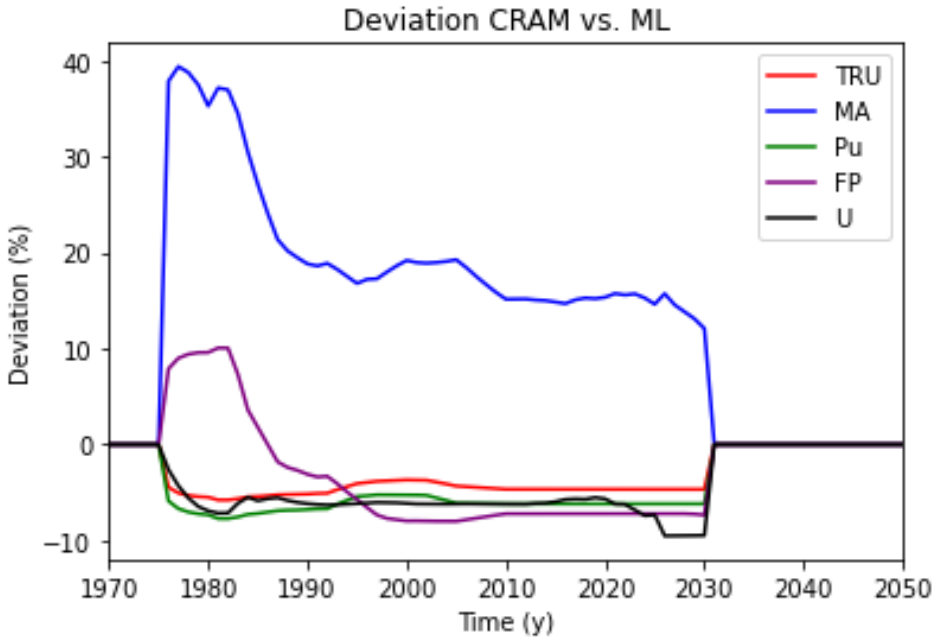


Figure 3-11: Relative deviation of the closed cycle scenario groups.

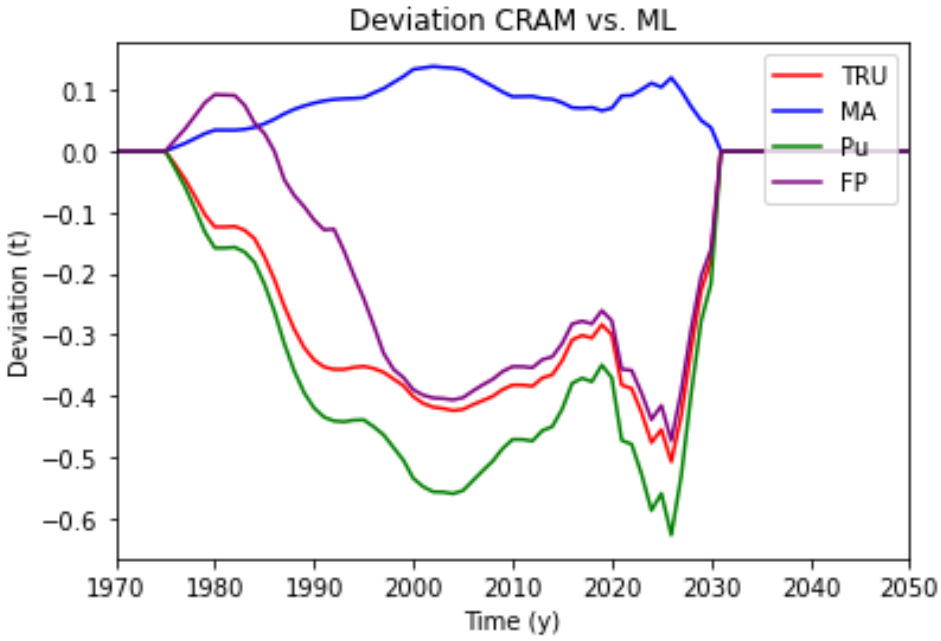


Figure 3-12: Absolute deviation of the closed cycle scenario groups.

One interesting aspect is the pattern change of deviation evolution between the open cycle and closed cycled scenario. These changes are induced by the scenario modeling, given that the timing for conventional nuclear power plants are not equal, so different rates of generation and/or consumption are developed across the whole runtime.

The deviations are steadily kept between +0.2 t and -0.6 t for the TRU, MA, total Pu and FP groups, and again less than 10 % of relative deviation for all the groups except for the MA one.

3.2.1. Resource Consumption Benchmark

To measure the resource consumption for each model (UOX and MOX) and to test it against the original model resource consumption, both the open and closed cycle scenarios were run 40 times each so the resource stats from the CPU could be retrieved and logged.

The results of these 40 runs were the following are presented in Figure 3-13.

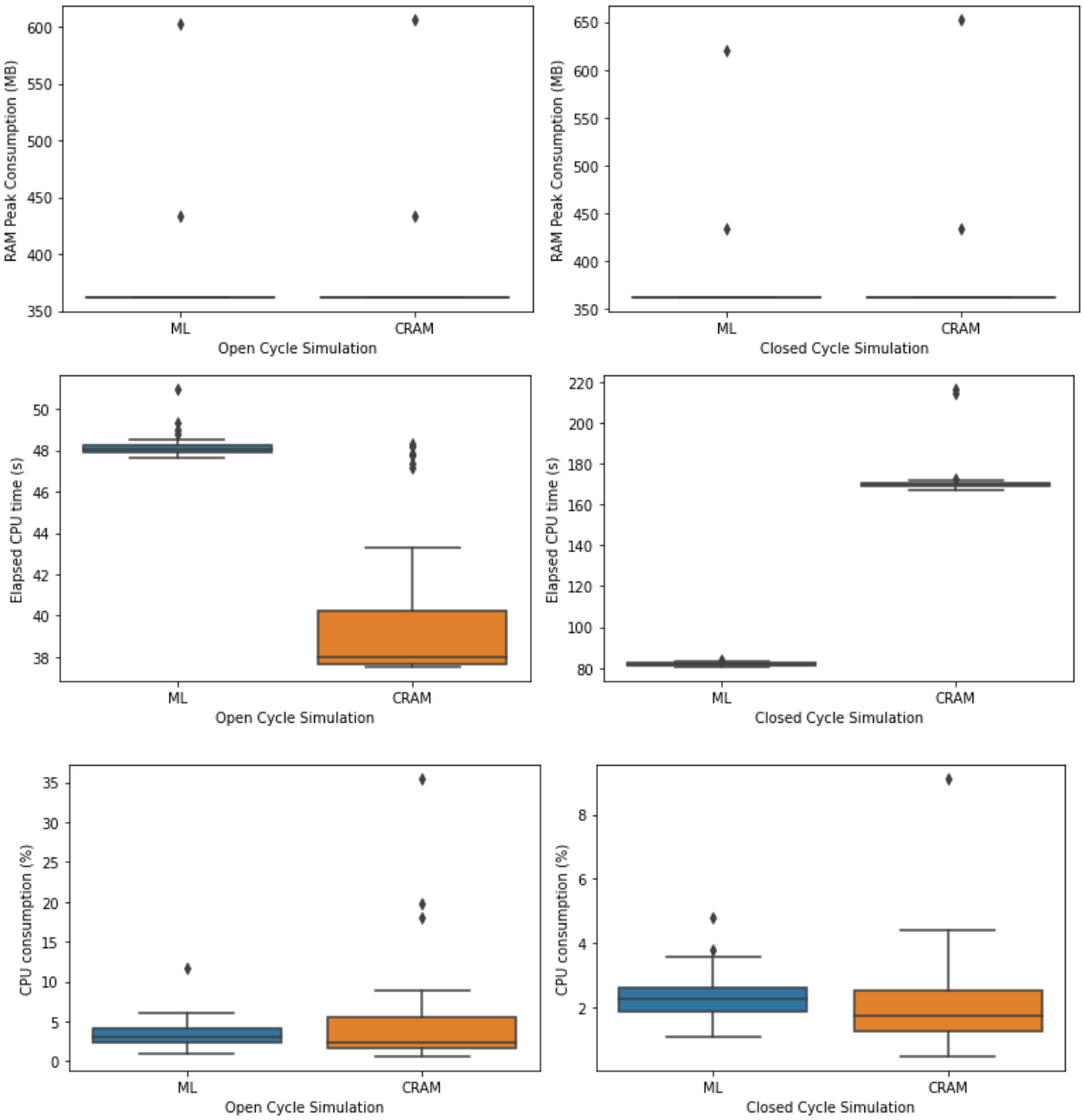


Figure 3-13: Results of the time benchmark for ANICCA.

It can be seen that no special enhancement has been observed for memory consumption, and even in the CPU consumption the CRAM seems to be a better performer for complex scenarios (e.g., the closed cycle one), nonetheless the resource consumption is even for both approaches (CRAM and ML).

On the other hand, the noticeable change was given in the execution time, especially for the closed cycle simulation, where the ML could finish the calculations up to 100 seconds earlier which is a huge share of the total time that takes for CRAM to finish.

However, a slight increase (over 8 seconds) of the elapsed time for the open cycle scenario was given for the ML, nonetheless, both memory and time consumption are strongly dependent on the designed program flow for the irradiation module of ANICCA.

In the case of the machine learner, the biggest share of the elapsed time was taken by the Keras Tensorflow module importing and initialization. It has also been noticed how embedding both the prediction and the load functions of the model in a loop dramatically affects the calculation time.

Taking the previous statements into account, it shall be said that the benchmark was not performed in the most optimal and favorable conditions, but in the one that the program flow could deal with both UOX and MOX prediction tasks, so an improvement margin is expected in both memory and elapsed time stats.

4. DISCUSSION

4.1. On the Neural networks performance

The results for both MOX and UOX models show good predictions in terms of overall error in the context of spent fuel characterization and fuel cycle analysis problems. Nonetheless, several biases and peculiarities of the model have been found.

Firstly, as a general remark, no great differences have been found between the error for MOX and for UOX models when looking at the tracked isotopes separately. Still, for certain isotopes in UOX such as the Pu-238 and Am-241, it is noticed how the predictions get biased for very low burnup values, and this error increases with the enrichment at this same burnup level reaching its maximum for 6% enriched UOX fuel. It is not a realistic scenario to have any of these isotopes in such quantities in almost fresh UOX fuel, at the same time is also not a realistic scenario to consider enrichment this high or discharge burnup values so small in LWRs, so the predictions at that point are out of the realistic bounds of the problem for spent fuel cycle analysis. These problems are related with the definition of relative error (in this case a coefficient of small values), causes the error to increase whilst having a reasonable absolute error. This is noticeable in low burnup values due to the fact that at these points, very little amount of such heavy isotopes is expected to be found in the fuel. In fact, at that point the relative error of all the isotopes (including Pu-238 and Am-241 in which the effect is more noticeable) is relatively high. Also, at the highest enrichment levels for UOX and MOX many error zones exist where the quantities are overestimated whatever the discharge burnup is. Consequently, it seems that the neural networks have two dead zones for both the lowest burnup and the highest enrichment values for both UOX and MOX models.

This can be explained taking into account that the error for low burnup values is greater for fission products than for fuel components, because at this point very low quantities of FP are built up in the core, so relative error tends to increase according to the previous mentioned effect. Indeed, the statistical dispersion is greater for lower burnup values according to the SERPENT2 uncertainty test, so it can be determined that the neural network is only replicating the uncertainty through the layers.

In the case of the high enrichment dead zone, the Nd-148 error has been taken as a reference providing that it is perfectly known that the Nd-148 amount is linear with the burnup (Figure 4-1), however, it still seems to have an error surge for high enrichment and high burnup values, this trend has been proven to exist in every tracked isotope except for the H-3. Knowing that this effect appears in every isotope plus the fact of not expecting deviation for Nd-148 for any enrichment levels, it is concluded that the error is due to a bias induced within the neural network's training set borderline values.

This first was thought to be a problem with saturation related with the data scaling (which usually uses to happen for input values close to the bounds of the activation function, which in this case is 0 to 1) but in this case, several isotopes have their maximum mass density values in other zones of the grid, so saturation was expected in other enrichment values. Besides, the output frequency has been checked in order to find out if saturation existed, but no high frequency could be associated with a certain value for any nuclide.

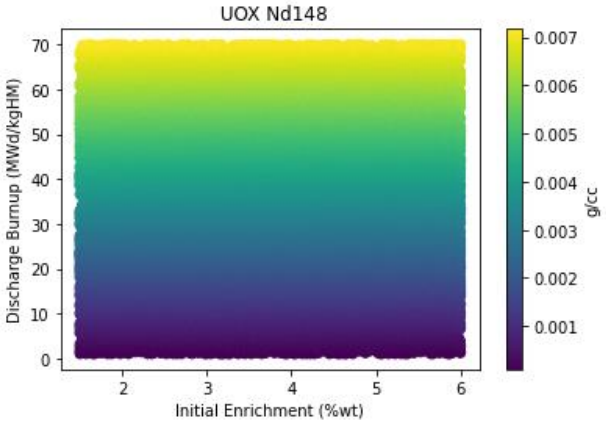


Figure 4-1: Nd-148 concentration as a function of BU and IE. Mass is not peaking at the upper right corner where the maximum error zone is.

However, it was found that the evolution rate of the mass density with the burnup for a given enrichment varies with the enrichment level in the original dataset, so it is concluded that those error structures are direct effects of the neural network adapting to all these secondary trends in the data. To illustrate this fact, in Figure 4-2 three predicted (orange) vs. actual (blue) plots for a supposed linear with burnup isotope such as Cs-137 are shown.

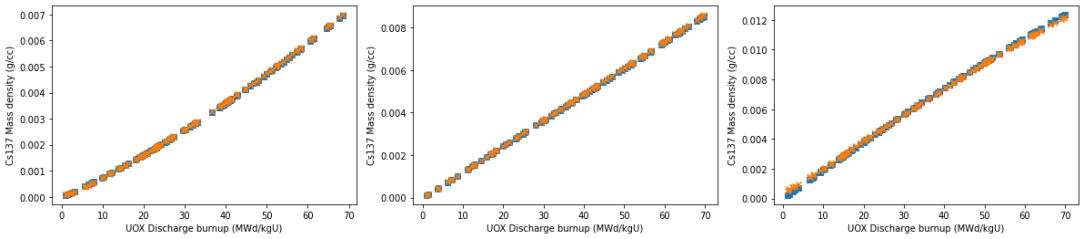


Figure 4-2: From left to right, Cs-137 mass density evolution for UOX enriched at 1.5%, 3% and 6% from left to right. Please note the different evolution of the trend and the inversion of the concavity when the enrichment increases.

Lastly, another neural network was employed to predict only the Cs-137 quantity instead of the 73 nuclides at the same time, and yet the same pattern persisted. Also, fearing a perturbation induced by the outliers, the neural network was retrained removing the isotopes with greater mass than the Cm245, but still the pattern persisted. So it was finally confirmed that these error zones are related with the dataset structure rather than with the neural network itself or a physical phenomenon, nonetheless, these errors are away from the average realistic values for PWRs. At the same time, it also shall be noticed that the error spikes in the decay steps are more likely to appear in certain isotopes than others (He-4, Y-90, Ru-106, Rh-106, Sb-125, Te-125m, Cs-133, Ce-144, Pm-147, …). Conversely these isotopes are very different from each other, ones are stable and others are radioactive. These nuclides are more affected by the error because are prone to be generated or removed during decay steps by engaging in different nuclear reactions ending up with two different mass densities for the discharge burnup values, so given that the neural network lacks a feature for distinguishing between pre and post decay densities, these error rises for these given intervals.

Nonetheless, the error is acceptable for this application. Besides, ANICCA lacks a proper input for the model to discriminate the decay step at the moment.

Except for the outliers encountered in the A vs Z tests, the fact of having low error levels for light isotopes and for the vast majority of heavy metals shall be remarked. Nevertheless the number of outliers increases with the enrichment for UOX. At the same time, no correlation between error and atomic mass has been observed, i.e., error for lighter isotopes is similar to the error for the heavy metals.

On the other hand, MOX has shown to have a nice average error without having many outliers (only Cf-251 for the highest enrichment). The fact of having better predicted isotopes could be because of the MOX having part of the chain of the heavy isotopes (which are the less accurate) already, as a share of the original plutonium vector from the very beginning. For instance, the original fresh MOX plutonium vector has Pu-242 and Am-241, which are fission precursors of Am-242, Am-243 and Am-244 (indirectly via Am-243). These americium isotopes are in the production chains of Cm-242, which decays in Cm-243 and Pu-238 which is also in great store in the MOX fuel in comparison with UOX fuel. Following the same logic, the Californium isotopes come from Curium isotopes via fission reaction, so it seems reasonable to accept this hypothesis.

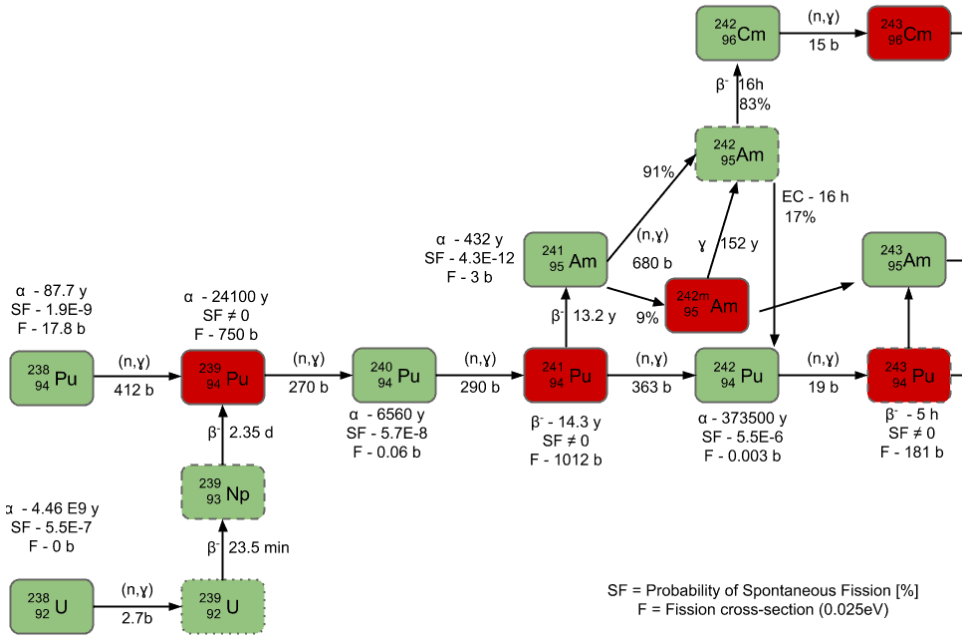


Figure 4-3: Fission path of Plutonium. Obtained in JANIS with JEFF-3.1.1. Library [53].

However, UOX has more outliers at high and medium enrichments. These outliers are those with an atomic mass greater than 246, curium and californium mainly. The problem with these isotopes is the huge error values for almost every grid point for both fuel types' models; nonetheless, according to Figure 3-4 most of these outliers get into acceptable limits at reasonable and usual enrichment and burnup values.

The Cf-251 is by far the worst overall predicted isotope due to several aspects at both computational and physical level. First it is important to notice that the relative error is several orders higher than average due to having very low quantities for heavy isotopes in almost every point of the grid, because it starts to appear at high burnup, so the neural networks have problems assessing whether the quantity is null at a certain point or not. This is a problem related to saturation, so the output frequency of the N.N. for the Cf-251 output was checked, as shown in Figure 4-4.

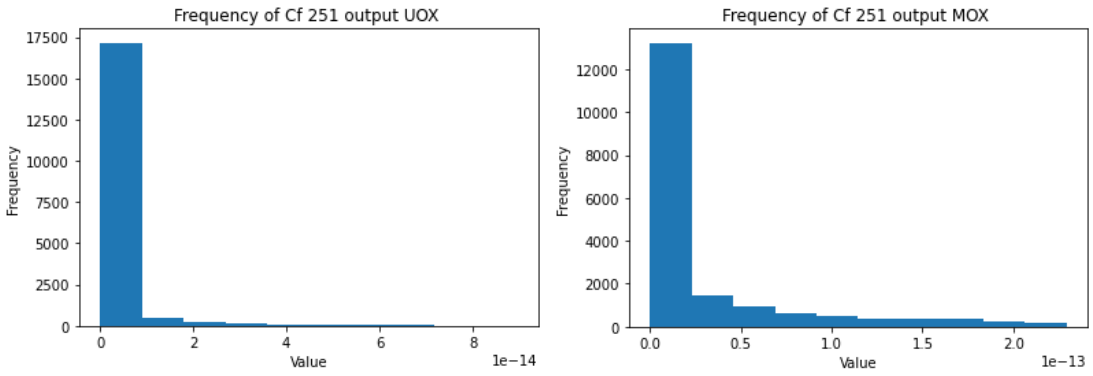


Figure 4-4: Frequency output for Cf-251 for both models. A high frequency in proportion with the other outputs reveals a saturation problem with near zero values [54].

The saturation is expected to happen when the neural network is facing nearly zero values (for medium to low burnups the Californium isotopes are around 1E-20 to 1E-33 in UOX data set). On the other hand, MOX fuel dataset has quantities for californium 5 orders greater, which is why it seems that the error for heavier isotopes improves. Indeed, if SERPENT2 uncertainties that were calculated in Section 2.4.1 are considered, the uncertainty for Cf-249 (which is also an outlier) is greater than 100% for both MOX and UOX datasets (Figure 2-17). Then, neural networks are just propagating the uncertainties.

This effect of saturation caused by error propagation is amplified by the relative error of minuscule quantities and has also been observed in the consulted reference for the deep learning approximation in the CYCLUS code [19].

Finally, by looking at the summary Table 3-2, it seems clear that the error for fuel cycle analysis is admissible for almost every point for the given enrichment and burnup ranges, taking into account that the models are being tested against SERPENT2 results and that the burnup and enrichment ranges are within realistic bounds.

Burnup range includes the average discharge burnups for PWRs, that are within 35 and 40 MWd/kgHM. It also includes the near to far future previsions on extending burnup further up to 70 MWd/kgHM, according to consulted references [55],[56].

Then, providing that both burnup and enrichment ranges that are within realistic bounds have a decent error without taking in consideration punctual outliers, it can be stated that both models are suitable for making predictions in fuel cycle analysis applications.

4.2. On the ANICCA´s benchmark

Both of the two benchmarks performed well in terms of relative error for every studied group, nonetheless, a greater deviation can be noticed for the minor actinides. The minor actinides are constantly overestimated for the total runtime. ANICCA takes into account the following nuclides from the studied list for the MA group: Th-228, Th-229, Th-230, Th-232, Pa-231, Am-241, Am-241m, Np-237, Cm-244, Cm-245, Cm-246, Cm-247, Cm-248, Cm-250, Bk-249, Cf-249, Cf-250 and Cf-251.

The nuclides for which the main absolute deviation given in mass is greater for every year are Np-237, Am-241 and Cm-244. Besides, the relative error averaged over every grid point (all burnup and enrichment values) for these isotopes extracted from the UOX neural network vs. SERPENT2 test were 0.25%, 1.80% and 1093.70% respectively.

Given that these error values are given with a sign (not absolute values) it implies that a constant overestimation bias can be found in the predictions of the neural network for UOX. However, for the MOX neural network, the errors for said isotopes are kept around -0.3%, which means that the overestimating trend is only significant for the UOX neural network. The recurring overestimation for isotopes with higher mass is affecting the MA group, so that is why these effects can be appreciated in both open (only UOX) and closed (UOX and MOX) cycles.

If the deviation for MA is checked, it can be seen how is kept constant for the open cycle while in the closed cycle scenario it starts decreasing after approximately 1982, which is exactly the year in that scenario when Doel 3 and Tihange 2 start operating. These power plants use MOX fuel, so the predictions start to be made with the MOX model which is more accurate for the nuclides belonging to the MA group, hence the deviation reduction. This same deduction can be made with the variation with time on the FP deviations.

On the other hand, and except for said effect, the deviation is kept constant when power plants are operating and then, it tends to zero when a positive slope is present in the inventory vs time plots. This is because once the models are imported to ANICCA, the predictions are not of a stochastic nature given that the model making the predictions was trained only once before deploying it. If the model was trained before every prediction (which would be impractical because of the high time consumption per prediction) the results would vary slightly. The fact of having the same values for CRAM and ML in the positive slope regions can be explained considering that these regions correspond with the time periods when the power plants are being refueled, which means that only fresh fuel is being employed and neither the CRAM nor the ML are being involved in said calculations, so masses tend to match in these periods.

When discussing the integration of the ML models with ANICCA, it can be stated that the fact of both approaches following the same evolution over time is a good indicator of a proper integration, nevertheless, when accuracy is taken into account, it is necessary to remark that the machine learner has been tested against SERPENT2 (which was used to generate both training and data sets) so, given the nature of the Monte Carlo simulations, the inventory calculations of SERPENT2 shall be taken as a reference. In that case, an overestimation that happens as a flaw of the ML can also be noticed as an overestimation in the CRAM vs. ML benchmark, thus, the CRAM approximation is not far from SERPENT2 estimations.

This implies that in terms of accuracy both methods are even and produce reasonable values for the overall studied group, even having slightly higher deviation values for certain individual nuclides. Then, the ML model has a good performance when predicting gross overall group

inventory or when predicting those nuclides that are not particularly biased by the effects discussed in Section 4.1.

In terms of resource consumption, there is not much of a difference between the required memory and CPU usage for CRAM and fro ML models. The performance of the total CPU time when solving complex case scenarios is important, and is substantially lower in the ML when solving closed cycles and advanced scenarios and not so different from the original version when solving open cycles and simpler scenarios.

The enhancement over computational time should be a clear advantage when facing complex scenarios) predicted to some extent in the discussion previously made to the execution of this work (Section 1.2), where how a complex case could make the total required computational time rise in significant numbers was discussed about. Also it should be stated that the fixed share of the total elapsed time which is taken by the Keras environment load and the dependent shares that accounts for the individual prediction time, are highly sensitive and dependent on the flow of the irradiation module script. This reflected in the testing of both scenarios separately, where under certain settings a total improvement on the elapsed time for both cycles was achieved, which was not the case on the resource benchmark because at the time being, the inputs for both UOX and MOX fuels could not be provided at the same time to the ML models in a way that program stability was granted. The reason why if because how ANICCA itself is coded, so maybe with further modifications of the program flow, a total enhancement on the prediction time could be achieved.

4.3. Work impact assessment

Given that the purpose of this thesis was merely the construction of a demonstrative solution with a relatively new technique in the nuclear field, no ostensible economic impact is foreseen, the last is also true for the environmental impact.

However, the successful deployment of the ML model together with the quality of the predictions, show the feasibility of integrating ML technology with conventional nuclear fuel cycle codes. At the same time, it shows that computational resources could be potentially spared while simplifying a substantial part of the codes (in this case, the CRAM devoted module).

Besides, the working principle of the ML techniques are simply based on data, which is getting more and more available these days. So maybe in the short run, these inventory production datasets could be retrieved from operational experience of the different power plants, so that more specific and dedicated models could be made for a reactor, considering as well the operating history (refueling periods, incidents, …). In the long run, it could be possible to establish non-supervised machine learners able to perform data crawling so they could be in constant training. At the same time, the thesis itself remarks the importance of the availability of reliable data that can only be obtained in a proper way from updated nuclear data libraries and accurate codes such as those based on Monte Carlo method. In a different point of view, a properly trained ML could overcome the need for burnup depletion calculations when the drawbacks of MC methods are not desired (long calculation time, …).

5. CONCLUSIONS

Every objective set in Section 1.5 was accomplished, which led to the total completion of the thesis and resulting in the total integration of the developed ML models in the irradiation module of ANICCA.

The training and data curation process demonstrated that a suitable dataset for a neural network of this nature can be obtained with well-known tools such as SERPENT2. At the same time, curation process was easily performed by taking advantage of the available resources in the Python environment.

Regarding the feature engineering, the cooling time-based discrimination was enough to satisfy the discrimination needs of the inventory. Nonetheless and for future developments based on this idea, it would have been desirable to provide the neural network with a third input that manages these cooling time issues by either assessing the cooling time itself or a derivate variable (pool time, refueling time, …). This could avoid the waste of certain burnups where the decay steps were performed, thus increasing the NN performance in prediction tasks. The problem with this modification would be to properly pass the third output through ANICCA to the models, because ANICCA is not currently providing any variable that can be used for this purpose.

In fact, both feature engineering and data curation processes are of crucial importance given the high time consuming SERPENT2 calculations required for the database generation. Detailing previously the needs to be covered by the neural network could suppose a huge difference in the data curation process and in the prediction results themselves. From a different perspective, obtaining enough data points could take up to several weeks, so it is desirable to do small scale tests and to foresee the feature needs before running calculations.

Subsequently, it is important to generate these datasets with a current and updated nuclear data library given that most of biases in the datasets are propagated from there. At the same time and as a lesson extracted from the bias assessment performed over the SERPENT2 outputs, is important to check that the different calculations made in SERPENT2 are not correlated and that no important deviations are present in the successive calculations for a given grid point (i.e. a certain IE and BU pair of values).

The optimization of both neural networks was achieved effortlessly with the built-in methods from Keras, even so, the optimal neural network architecture search can be explored deeper, specially taking into account that no benchmark was performed over the different architectures, so, given that the neural networks with architecture based on the hyperband optimizers were enough to suffice the project needs, no more development over the optimization part of the project was conducted.

The results of both models are good in terms of relative error and for the average burnup and enrichment values for both MOX and UOX datasets. Even so, many outliers can be detected at borderline values of burnup or enrichment. Nonetheless, these values are not realistic so, at the same time, they are not prone to induce serious error in real life simulations. Nevertheless, the error has been properly documented for both models.

It is interesting to remark the fact of the MOX neural network having a poorer performance measured in terms of the loss function has not affected the MOX prediction capabilities of the ML approach in ANICCA, which is mainly because the total loss is quantified for every grid

point being them rather realistic of unrealistic, but at the same time, the vast majority of real case burnup and enrichment values are within good error levels.

Regarding the integration of the models in ANICCA, it can be summarized that their performance is good in resource consumption terms besides of maintaining a deviation with the original CRAM method. In fact, the resource consumption could be optimized even more by restructuring certain parts of the program as well as by employing certain techniques that will be discussed in the future works chapter.

It shall be noticed that the deviations over the total mass inventory for the studied groups are not significant for many applications of the fuel cycle codes. Fuel cycle codes are estimative tools that provide gross and high level simulations for a certain scenario, so a deviations on the level of the ones analyzed are within the assumable uncertainty of the simulation by definition. Besides, these uncertainties could be properly addressed by taking into account the economic impact of them in the fuel cycle reprocessing and final disposal efforts, tasks are currently under development for ANICCA code.

On the other hand, the neural network approximation lacks the versatility that CRAM has, this is due to the rigid dataset-training-deployment-prediction staged process the neural network design has to follow, for instance, currently and as it was explained before, this MOX neural network can only predict, within reasonable accuracy levels, the final inventory of a given pre-fixed plutonium vector, that is because the datasets were generated following that plutonium vector, so if a new MOX composition is used or if a greater storage time is reached (which implies the buildup of Am-241) the neural network is less likely to provide an accurate prediction. While it is true that also a new ANICCA old library would be required when the Pu vector changes, this calculations are not so time consuming as the needed for setting up a new neural network model.

In the same aspect of rigidity, these neural networks can only predict the total mass of up to 73 isotopes, so if more isotopes are needed in the list, again a new neural network (or a complementary one) will be needed. At the same time, the quantity of predictable isotopes is not itself limited but as it has been seen, certain isotopes are not well normalized together with the rest given that they are not in the same amount, and this leads to saturation problems. Besides, short lived isotopes are not well predicted if a smaller scale of time is introduced as an input of the neural network (such as cooling times).

To summarize, the neural network approach, while being a more rigid solution when compared to CRAM, it can be as accurate as CRAM with the proper choice of architecture and hyperparameters. Besides it almost eliminates the dependency on third party libraries, like those ANICCA's previous model was using, so a simplification on program flow can be made. Also, there is no need to obtain indirect data from interpolation between the values of said libraries when a neural network that has been trained with a dense dataset of realistic grid points is available, thus increasing resolution of the data available for ANICCA. Also, speeding the calculation process can be a huge benefit in certain applications for complex scenarios.

As a drawback, the deviation can be higher due to a poor performance of the prediction task over certain important isotopes for some groups, but this problem could be addressed by a finer neural network with more features and a proper curated dataset, besides, in most cases, this deviation is not time dependent, so it can be removed from the calculations given that the bias is constant. Another drawback would be the rigid nature of a neural network, so, this approach could be considered halfway between a slow and precise calculation such as those produced with SERPENT2, and the quicker choice of the original irradiation module of ANICCA.

6. FUTURE WORK

As a result of the conclusions of the thesis, further developments could be performed on uncertainty mitigation from the ML model, removing the stiffness of the current NNs models, and on reducing the total elapsed time of said models in ANICCA.

6.1. Mitigating uncertainties of the ML model

This can be addressed by using several approaches. The first could be including the aforementioned third input to the neural network which would take into account the decay times for short lived isotopes and the refueling periods when no irradiation processes are being carried away.

In that case, SERPENT2 outputs a double pair of IE and BU values with different mass densities for each pair, the first one is the inventory before decay step and the second one is the inventory corresponding to the simulation after the decay step, so in terms of feature engineering it is simple to discriminate these two different rows of the dataset by assigning them with a 0 or 1 value meaning the first that the data is from a previous stage to the decay step and the second that is from after that decay step. Also, a different set of values could be studied if saturation become an issue when training the models (maybe if the dataset is being scaled from 0 to 1, having that binary-like values affect the activation of certain neurons).

At the same time, performing feature engineering over the problematic nuclides (californium, curium, …) could shed some light on which values are more sensible for the forecasting of said isotopes. If that is done, a secondary dedicated neural network could be designed to cover the prediction of those isotopes and the original model could be tuned thanks to that process.

The second approach, that can be considered simultaneously, comprehends the use of the error obtained by the evaluation of the neural network performance method, given that the predict function of a model is deterministic, (i.e., every prediction with the same inputs is the same), the error is constant so it can be taken into account when making prediction, thus controlling the uncertainty for each dimension of the output. This can also be done by directly observing the deviation when the model has been already integrated in ANICCA or in another host.

Finally, more tools and strategies can be used when assessing the neural network performance, such as the K-fold cross-validation technique. In that way the neural network trains in difference sequences and in these sequences every test and training bundles are changed, so a more solid indicator against overfitting can be used. Besides with this method, the total set of data can be input for the neural network lacking the need of splitting the total dataset into training and testing set, which implies an increase on data's usability.

6.2. Design of a general-purpose neural network for UOX and MOX fuels

This work could be carried away along the previous one, the main objective would be to obtain more outputs parameters besides the isotopic inventory, such as activity and generated heat, at the same time, and for the MOX neural network, the input dimension could be reshaped so instead of taking only discharge burnup values and IPC, could take every Pu isotope weight percentage or fraction for fresh MOX fuel. This will allow the neural network to forecast the final inventory for every MOX fuel type regardless the initial plutonium vector. At the same time, or alternatively, the average storage time could be taken included as input to account for Am-241 production via Pu-241 decay.

The main obstacle to overcome here would be the prohibitive computational time, it shall be reminded that for this dataset, a total of 10 nuclides were inserted as composition for SERPENT2 (Pu, Am, U and O) so, besides burnup, this will make up to 11 inputs to provide the neural network with, so for obtaining the same grid points of the current dataset for ten different Pu vectors, it will imply a total estimated calculation time of 10 weeks if the same burnup points that were taken here are used (241 steps). For overtaking this task, HPC services would be useful.

6.3. Model deployment in ANICCA for a faster calculation runtime

The key aspect of a faster calculation is the deployment of the neural network into the environment of ANICCA, mainly because of two processes: the prediction time, which is dependent on the location of the predict instruction of the model in the code (for example, this time is incremented when the instruction is called from a loop such as for or while), and the Keras environment loading time. Besides choosing cautiously the indentation level of the instructions, there are several techniques that can serve this purpose.

The current trained models are saved in the form of a hdf5 file. This kind of files is large to store and slow when performing the loading, so a technique for speeding up this process called graph freezing can be applied.

Storing a static model produces a unique file containing both neural network's architecture and checkpoint variables, the hyperparameters are stored within the graph structure, this way additional and unnecessary information for inference is discarded. The conversion converts the model from a hdf5 file to a protocol buffer file (.pb) so now the model can be called and used for inference in the same way as the previous one, only requiring Tensorflow to be imported to ANICCA.

Other solution could be building a deep neural network without a framework, which means that no API is involved, and it would only be coded in Python, this, when following the steps of the neural networks' working principle, is equivalent to use a pre-built API model with the advantage of not requiring any additional module to be imported. Though, here it is important to take into account the tradeoff between saved time and model complexity, because in many cases, saving a few seconds will not be worth the effort.

7. REFERENCES

[1] IEA, “Nuclear Power in a Clean Energy System,” 2019. doi: 10.1787/fc5f4b7e-en.

[2] Oecd, Nea, and J. Hill, “Public Attitudes To Nuclear Power.,” in *Annual International Conference - Canadian Nuclear Association*, 1981, no. 6859, pp. 28–31, [Online]. Available: https://www.iea.org/reports/nuclear-power-in-a-clean-energy-system.

[3] C. Fernández Álvarez and M. Gergely, “What is behind soaring energy prices and what happens next?,” *IEA, Paris*, 2021. https://www.iea.org/commentaries/what-is-behind-soaring-energy-prices-and-what-happens-next.

[4] E. TESIO, I. CONTI, and G. CERVIGNI, “High gas prices in Europe : a matter for policy intervention?,” *Policy Briefs, 2022/06, Florence Sch. Regul. [Gas]*, vol. 06, pp. 1–7, 2022, [Online]. Available: https://cadmus.eui.eu//handle/1814/73596.

[5] S. Brown, “Soaring fossil gas costs responsible for EU electricity price increase,” *Ember Climate*, 2021. https://ember-climate.org/commentary/2021/10/14/soaring-fossil-gas-costs-responsible-for-eu-electricity-price-increase/.

[6] J. Jewell *et al.*, “Comparison and interactions between the long-term pursuit of energy independence and climate policies,” *Nat. Energy*, vol. 1, no. 6, p. 16073, 2016, doi: 10.1038/nenergy.2016.73.

[7] L. Rodríguez-Fernández, A. B. F. Carvajal, and V. F. de Tejada, “Improving the concept of energy security in an energy transition environment: Application to the gas sector in the European Union,” *Extr. Ind. Soc.*, no. xxxx, 2022, doi: 10.1016/j.exis.2022.101045.

[8] European Commission, “EU Taxonomy: Commission presents Complementary Delegated Act to accelerate decarbonisation,” *Eur. Comm.*, no. February, 2022.

[9] L. M. Pierpoint, “A Decision Analysis Framework for the U . S . Nuclear Fuel Cycle by Massachusetts Institute of Technology , 2008 Submitted to the Engineering Systems Division in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy at the Massach,” 2011.

[10] R. J. Finch, Y. Hwang, and S. M. Deland, “a Comparison of Simulators for Nuclear Fuel Cycle Needs in East Asia,” 2010.

[11] C. Poinssot *et al.*, “Assessment of the environmental footprint of nuclear energy systems. Comparison between closed and open fuel cycles,” *Energy*, vol. 69, pp. 199–211, May 2014, doi: 10.1016/J.ENERGY.2014.02.069.

[12] SCK-CEN and B. L. Sjenitzer, “EXTERNAL REPORT Analysis of the Belgian Nuclear Fuel Cycle Using the ANICCA code,” Boeretang 200 BE-2400 Mol Belgium, 2014. [Online]. Available: http://www.sckcen.be.

[13] I. M. Rodríguez, A. Hernández-Solís, N. Messaoudi, and G. Van den Eynde, “The nuclear fuel cycle code ANICCA: Verification and a case study for the phase out of Belgian nuclear power with minor actinide transmutation,” *Nucl. Eng. Technol.*, vol. 52, no. 10, pp. 2274–2284, Oct. 2020, doi: 10.1016/J.NET.2020.04.004.

[14] M. Pusa, “Rational approximations to the matrix exponential in burnup calculations,” 2011.

[15] M. (VTT) Pusa, *Numerical methods for nuclear fuel burnup calculations*. VTT, 2013.

[16] A. Stankovskiy, G. Van Den Eynde, P. Baeten, C. Trakas, P. M. Demy, and L. Villatte, *ALEPH2 - A general purpose Monte Carlo depletion code*. United States: American Nuclear Society - ANS, 2012.

[17] B. Sjenitzer, “Anicca Documentation. Release 1.0.” SCK-CEN, 2014.

[18] H. Li, “Which Machine Learning Algorithm Should I Use?,” *KD nuggets*, 2017. https://www.kdnuggets.com/2017/06/which-machine-learning-algorithm.html.

[19] J. W. Bae, A. Rykhlevskii, G. Chee, and K. D. Huff, “Deep learning approach to nuclear fuel transmutation in a fuel cycle simulator,” *Ann. Nucl. Energy*, vol. 139, p. 107230, May 2020, doi: 10.1016/J.ANUCENE.2019.107230.

[20] B. Leniau *et al.*, “A neural network approach for burn-up calculation and its application to the dynamic fuel cycle code CLASS,” *Ann. Nucl. Energy*, vol. 81, pp. 125–133, 2015, doi: 10.1016/J.ANUCENE.2015.03.035.

[21] M. El-Sefy, A. Yosri, W. El-Dakhakhni, S. Nagasaki, and L. Wiebe, “Artificial neural network for predicting nuclear power plant dynamic behaviors,” *Nucl. Eng. Technol.*, vol. 53, no. 10, pp. 3275–3285, Oct. 2021, doi: 10.1016/J.NET.2021.05.003.

[22] F. Bre, J. M. Gimenez, and V. D. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using artificial neural networks,” *Energy Build.*, vol. 158, no. November 2017, pp. 1429–1441, 2018, doi: 10.1016/j.enbuild.2017.11.045.

[23] A. Obuchowski, “Understanding neural networks 1: The concept of neurons,” *Becoming Human*, 2019. https://becominghuman.ai/understanding-neural-networks-1-the-concept-of-neurons-287be36d40f.

[24] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, doi: 10.1016/0893-6080(89)90020-8.

[25] A. Pinkus, “Approximation theory of the MLP model in neural networks,” *Acta Numer.*, vol. 8, pp. 143–195, Jan. 1999, doi: 10.1017/S0962492900002919.

[26] C. M. Bishop, “Linear Models for Classification,” in *Pattern Recognition and Machine Learning*, Springer, 2021, pp. 198–144.

[27] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” no. 1, pp. 2–8, 2018, [Online]. Available: http://arxiv.org/abs/1803.08375.

[28] Z. Waszczyszyn, “Fundamentals of Artificial Neural Networks,” in *Neural Networks in the Analysis and Design of Structures*, 1999, pp. 1–51.

[29] C. M. Bishop, “Neural Networks,” in *Pattern Recognition and Machine Learning*, .

[30] “How Does the Gradient Descent Algorithm Work in Machine Learning?” https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/ (accessed Mar. 04, 2022).

[31] “Setting the learning rate of your neural network.” https://www.jeremyjordan.me/nn-learning-rate/ (accessed Mar. 04, 2022).

[32] K. You, M. Long, J. Wang, and M. I. Jordan, “How Does Learning Rate Decay Help Modern Neural Networks?,” Aug. 2019, Accessed: Feb. 18, 2022. [Online]. Available: https://arxiv.org/abs/1908.01878v2.

[33] W. K. Follow, “Overftting vs . Underftting : A Complete Example,” *Towar. Data Sci.*, pp. 1–12, 2019.

[34] H. K. Jabbar and R. Z. Khan, “Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study),” pp. 163–172, 2015, doi: 10.3850/978-981-09-5247-1_017.

[35] Z. Elter, L. P. Balkeståhl, E. Branger, and S. Grape, “Pressurized water reactor spent nuclear fuel data library produced with the Serpent2 code,” *Data Br.*, vol. 33, p. 106429, Dec. 2020, doi: 10.1016/J.DIB.2020.106429.

[36] J. Leppänen, “Serpent – a Continuous-energy Monte Carlo Reactor Physics Burnup Calculation Code,” *Tech. Res. Cent. Finl.*, p. 157, 2011.

[37] R. Rossa and A. Borella, “Development of the SCK CEN reference datasets for spent fuel safeguards research and development,” *Data Br.*, vol. 30, Jun. 2020, doi: 10.1016/j.dib.2020.105462.

[38] “sklearn.preprocessing.MinMaxScaler — scikit-learn 1.0.2 documentation.” https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (accessed Mar. 04, 2022).

[39] G. Žerovnik *et al.*, “Observables of interest for the characterisation of Spent Nuclear Fuel,” 2018. doi: 10.2760/418524.

[40] S. Piet, “Selection of Isotopes and Elements for Fuel Cycle Analysis,” *Am. Nucl. Soc. - 4th Top. Meet. Adv. Nucl. Fuel Manag. 2009, ANFM IV*, vol. 1, 2009.

[41] P. Paviet-hartmann, W. Kerlin, and B. Steven, “Treatment of Gaseous Effluents Issued from Recycling – A Review of the Current Practices and Prospective Improvements,” *NEA/OECD Work. Actin. Fission Prod. partioning Transmutat. Exch. Meet.*, 2010.

[42] IAEA, “Management of Waste Containing Tritium and Carbon-14,” 2001.

[43] L. Ichou, R. ; Leclaire, N; Leal, “BURN-UP CALCULATIONS USING VESTA 2 . 2 ON THE UAM PIN CELL BENCHMARK,” 2021.

[44] L. Fiorito *et al.*, “On the use of criticality and depletion benchmarks for verification of nuclear data,” *Ann. Nucl. Energy*, vol. 161, p. 108415, Oct. 2021, doi: 10.1016/J.ANUCENE.2021.108415.

[45] IAEA, “Technical Report Series no. 415. Status and Advances in MOX Fuel Technology , International Atomic Energy Agency.,” Vienna, 2003. [Online]. Available: https://www.iaea.org/publications/6562/status-and-advances-in-mox-fuel-technology.

[46] A. E. Johnson, D. Kotlyar, S. Terlizzi, and G. Ridley, “serpentTools: A Python Package for Expediting Analysis with Serpent,” *Nucl. Sci. Eng.*, vol. 194, no. 11, pp. 1016–1024, Nov. 2020, doi: 10.1080/00295639.2020.1723992.

[47] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, 2018.

[48] J. Wang, J. Xu, and X. Wang, “Combination of Hyperband and Bayesian Optimization for Hyperparameter Optimization in Deep Learning arXiv : 1801 . 01596v1 [cs . CV] 5 Jan 2018.”

[49] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014, [Online]. Available: http://arxiv.org/abs/1412.6980.

[50] S. J. Reddi, S. Kale, and S. Kumar, “ON THE CONVERGENCE OF ADAM AND BEYOND.”

[51] J. Schmidt-Hieber, “Nonparametric regression using deep neural networks with relu activation function,” *Ann. Stat.*, vol. 48, no. 4, pp. 1875–1897, 2020, doi: 10.1214/19-AOS1875.

[52] A. Ozgur and F. Nar, “Effect of Dropout layer on Classical Regression Problems,” *2020 28th Signal Process. Commun. Appl. Conf. SIU 2020 - Proc.*, no. October, 2020, doi: 10.1109/SIU49456.2020.9302054.

[53] N. Soppera, M. Bossant, and E. Dupont, “JANIS 4: An improved version of the NEA Java-based nuclear data information system,” *Nucl. Data Sheets*, vol. 120, pp. 294–296, 2014, doi: 10.1016/j.nds.2014.07.071.

[54] A. Rakitianskaia and A. Engelbrecht, “Measuring saturation in neural networks,” in *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, 2015, pp. 1423–1430, doi: 10.1109/SSCI.2015.202.

[55] E. H. Uguru, S. F. A. Sani, M. U. Khandaker, M. H. Rabir, and J. A. Karim, “A comparative study on the impact of Gd2O3 burnable neutron absorber in UO2 and (U, Th)O2 fuels,” *Nucl. Eng. Technol.*, vol. 52, no. 6, pp. 1099–1109, Jun. 2020, doi: 10.1016/j.net.2019.11.010.

[56] “Impact of extended burnup on the nuclear fuel cycle INTERNATIONAL ATOMIC ENERGY AGENCY,” 1993.

8. TIME AND RESOURCES PLANNING

In the following section are related the different milestones in the thesis' project describing both the different tasks and duration per task, also a Gantt chart is included as an overview of the project management.

Table 8-1: Time planning for the whole project.

Id	Level	Name	Start	Due	Duration	Working Days
1	1	**Machine learning concepts**	01/17/2022	01/28/2022	12	10
2	2	Pandas' environment	01/17/2022	01/20/2022	4	4
3	2	Keras & Tensorflow environment	01/17/2022	01/28/2022	12	10
4	2	Neural networks theoretical concepts	01/17/2022	01/28/2022	12	10
5	1	**Preliminary Research**	01/31/2022	02/08/2022	9	7
6	2	Neural network prototype for Uppsala datasets	01/31/2022	02/04/2022	5	5
7	2	Output prototype post-processing	02/04/2022	02/08/2022	5	3
8	2	Neural network prototype from SCK: feature discrimination	02/04/2022	02/08/2022	5	3
9	2	Neural network re-training without Cooling Time	02/07/2022	02/08/2022	2	2
10	1	**Database Generation**	02/09/2022	03/17/2022	37	27
11	2	SERPENT2 Uncertainties check	02/09/2022	02/20/2022	12	8
12	2	Observable Selection	02/10/2022	02/17/2022	8	6
13	2	SERPENT2 input design	02/21/2022	03/02/2022	10	8
14	2	SERPENT2 UOX Calculation	03/03/2022	03/09/2022	7	5
15	2	SERPENT2 MOX Calculation	03/10/2022	03/16/2022	7	5
16	2	Data visualization & curation	03/10/2022	03/17/2022	8	6
17	1	**ML Integration In ANICCA**	03/02/2022	04/01/2022	31	23
18	2	Study of Documentation	03/03/2022	03/31/2022	29	21
19	2	Reverse engineering on Linux environment	03/10/2022	03/17/2022	8	6
20	2	Flowchart deduction	03/11/2022	03/15/2022	5	3
21	2	Interviews with previous developer	03/13/2022	03/29/2022	17	12
22	2	First implementation & testing	03/17/2022	03/22/2022	6	4
23	2	Definitive implementation	03/28/2022	03/31/2022	4	4
24	2	Portability	04/01/2022	04/01/2022	1	1
25	1	**Neural Network design**	03/10/2022	03/25/2022	16	12
26	2	Hyperparameter Tuner selection	03/10/2022	03/17/2022	8	6
27	2	Hyperparameter tuning for UOX	03/18/2022	03/21/2022	4	2
28	2	Hyperparameter tuning for MOX	03/22/2022	03/24/2022	3	3
29	2	UOX Neural Network training	03/23/2022	03/23/2022	1	1
30	2	MOX Neural Network training	03/24/2022	03/24/2022	1	1
31	2	Post-processing and model serialization	03/24/2022	03/25/2022	2	2
32	1	**Benchmarking**	04/05/2022	04/22/2022	18	14
33	2	Selection of an Opt. Scenario for UOX	04/10/2022	04/11/2022	2	1
34	2	Selection of an Opt. Scenario for MOX	04/12/2022	04/14/2022	3	3
35	2	Testing	04/15/2022	04/15/2022	1	1
36	2	Post-processing	04/15/2022	04/22/2022	8	6

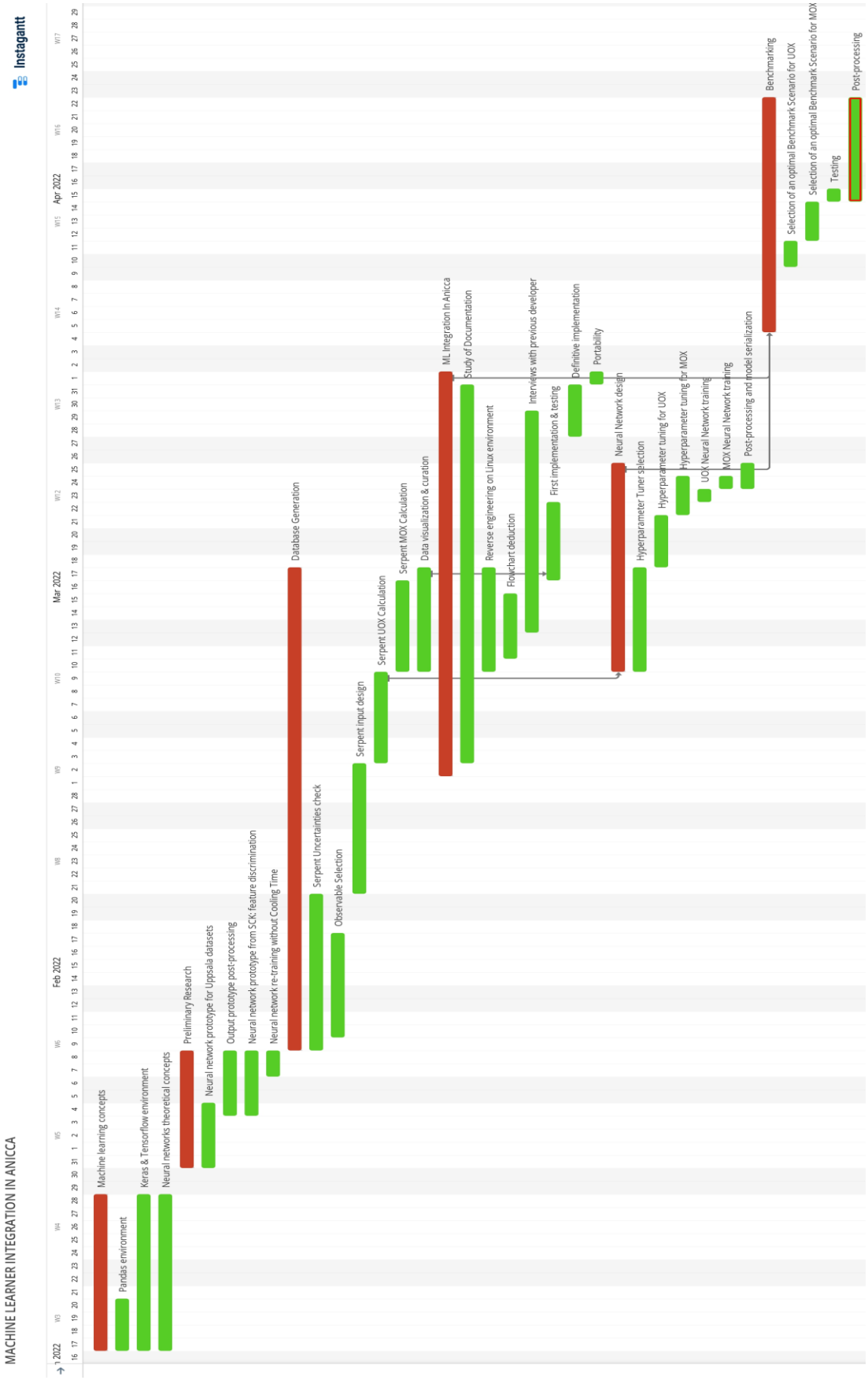


Figure 8-1: Gantt Chart.

This project was developed entirely under an internship in the Belgian Nuclear Research Centre and lasted from the 17th of January to the 22nd of April of 2022, lasting a total of 96 calendar days or 68 business days.

Regarding to the resources employed for the thesis development the following table containing related information is shown:

Table 8-2: List of software resources employed in the project.

Coding and multipurpose tools		
Name	**Description**	**Observations**
Anaconda	Python distribution	Free License
Spider environment	Part of Anaconda suite	Free License
Neural network design		
Name	**Description**	**Observations**
Tensorflow	Machine Learning & Artificial intelligence library	Free and Open-Source software library
Keras	Neural Network API	Open Source
Neural network architecture visualization		
Name	**Description**	**Observations**
Netron	.h5 models' visualization	Open Source
Database generation		
Name	**Description**	**Observations**
SERPENT 2.1.32	Continuous-Energy Monte Carlo Particle Transport Code	Under VTT license
Database pre & post-processing, data visualization & cross-validation		
Name	**Description**	**Observations**
Scikit Learn	Machine Learning API	Open Source
Pandas	Data analysis & manipulation	Open Source
Excel™ Spreadsheet	Data visualization	MS Office
ANICCA Testing tools		
Name	**Description**	**Observations**
ANICCA v1.0	Fuel cycle code	In-house Code
Oracle VM VirtualBox	Virtual machine software	Free License
Ubuntu	GNU-Linux distribution	Open Source

9. BUDGETING

The Budget for the project can be deduced out of the time planning and resources employed during the development, this budget can be divided into resources and services.

For the resources item, two categories have been included: Hardware and Software:

Table 9-1: Resources Budget.

Hardware Budget				
Concept	***Price (€)***	***Amortization[10]***	***Use***	***(1) Subtotal (€)***
Lenovo ® Thinkpad ® E15 Gen 3 (15'' AMD)	854.05	72 months	3.2 months	37.96
DELL P2217H Monitor	254.69	72 months	2.8 months	9.91
Lenovo Essential Mouse	29.25	72 months	3.2 months	1.30
Lenovo SK 8823 Keyboard	25.90	72 months	3.2 months	1.15
Software Budget				
Concept	***Price (€)***	***Amortization***	***Use***	***(2) Subtotal (€)***
SERPENT2	-	-	-	-
MS® Office® Professional 2021	579.00	-	-	579.00
Other SW	29.25	-	-	0.0
TOTAL FOR HW & SW (1) + (2)				***629.32 €***

For the services budget, the items include three parties for the human resources: the author of the thesis as a junior engineer, and both mentor and co-mentor as senior engineers, if salaries are then prorated, it can be extracted the hourly cost for each party, making 4.68€/hour for the junior engineer and 35 €/hour for both the mentor and co-mentor.

For the computational cost of the SERPENT2 calculations it has been assumed the costs of the cluster to be similar as those that exist for the Genius HPC facility for KU Leuven, the price per 1,000 credits for academic European projects is 3.5€ with a minimum purchase of 5,000 credits, and using the most basic node architecture requires 10 credits per node-hour plus a 0.84€ fee per node-day (if the node is used more than 24 hours).

Taking into account that with one node was used for a calculation that lasted for approximately 336 hours or 14 days, then the total price of the calculations has been 11.76€ for the continued run plus the price of 5,000 credits given that the minimum of credits was not reached, so a total of 29.26 € is estimated.

[10] Supposed Linear, no residual value after complete amortization.

Table 9-2: Services Budget.

Concept	Hours	Hourly price (€/hr)	(3) Subtotal (€)
Student	744	4.68	3481.92
Mentor	100	35	3500.00
Co-mentor	20	35	700.00
HPC services	336	-	29.26
TOTAL FOR SERVICES (3)			**7711.18**

The working hours for the author have been deducted from the total duration of the main projects (level 1 in Gantt chart) in some sort of correction to cover the working time during weekends and other variations.

Then the total budget for this project is (1) + (2)+(3): 8340.50 € VAT incl.

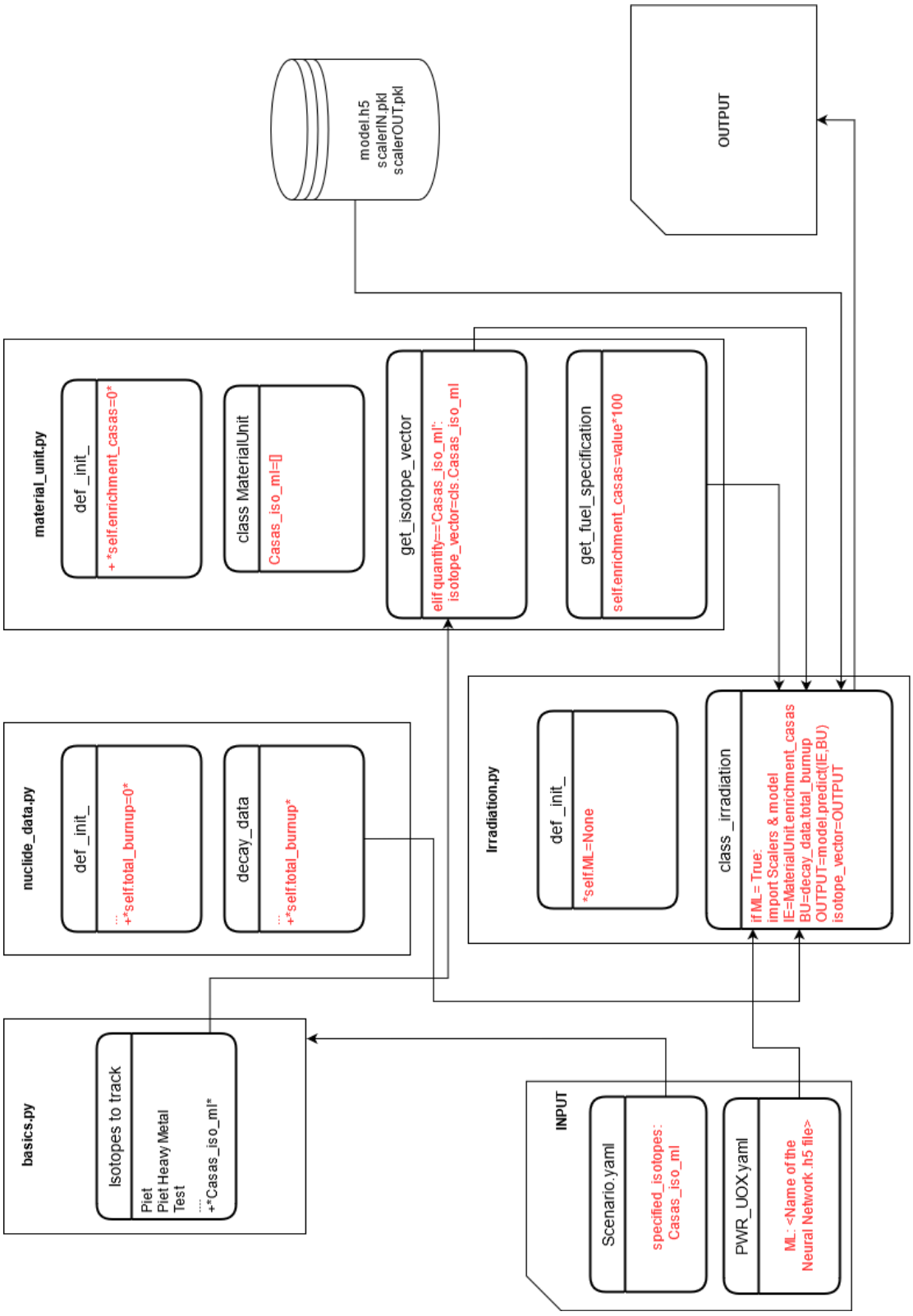
10. LIST OF FIGURES

11. LIST OF TABLES

12. LIST OF ABBREVIATIONS

ADS	Accelerator Driven System
ANN	Artificial Neural Network
API	Application Programming Interface
BOC	Beginning Of Cycle
BU	Burnup
CRAM	Chebyshev Rational Approximation Method
CT	Cooling Time
ENDF	Evaluated Nuclear Data File
EU	European Union
FP	Fission Product or Fission Products
HPC	High Performance Computing
I/O	Input / Output
IAEA	International Atomic Energy Agency
IE	Initial Enrichment
IPC	Initial Plutonium Content
JRC	Joint Research Centre
LEU	Low Enriched Uranium
MA	Minor Actinides
MC	Monte Carlo
ML	Machine Learner or Machine Learning
MOX	Mixed Oxide fuel
MOXEUS	Mixed Oxide fuel On Enrichment Uranium Support
MSE	Mean Squared Error
NN	Neural Network
PWR	Pressurized Water Reactor
ReLU	Rectified Linear Unit
SCK-CEN	Belgian Nuclear Research Centre
SNF	Spent Nuclear Fuel
TRU	TRans Uranium elements
UOX	Uranium Oxide fuel

APPENDIX A: Modifications in the original ANICCA code



APPENDIX B: Error distribution for tracked isotopes

Percentile 80 and 100 for each isotope, the values are relative error in percentage. Burnup is in MWd/kgHM.

	UOX (3-5%)						MOX (5-9%)					
Burnup	25-38		38.1-50		50.1-70		25-38		38.1-50		50.1-70	
Percentile	80	100	80	100	80	100	80	100	80	100	80	100
H3	9.40	39.24	7.66	28.01	6.59	17.82	10.03	27.20	7.51	21.06	6.30	15.69
He4	0.78	1.80	0.65	1.78	0.37	1.45	0.69	2.57	0.68	2.00	0.44	1.36
O16	3.41E-4	6.44E-4	3.54E-4	9.19E-4	3.40E-4	7.02E-4	3.4E-4	6.21E-4	3.4E-4	7.1E-4	3.5E-4	1.2E-3
Se79	0.25	0.59	0.19	0.41	0.17	0.37	0.29	0.96	0.22	0.72	0.16	0.73
Kr81	1.75	3.71	1.08	2.25	0.48	1.48	0.60	1.84	0.35	1.32	0.26	0.79
Kr85	0.25	0.60	0.15	0.62	0.14	0.62	0.20	0.71	0.22	0.70	0.15	0.92
Sr90	0.25	0.51	0.16	0.40	0.14	0.40	0.25	0.76	0.20	0.65	0.15	0.76
Y90	0.50	2.99	0.31	3.42	0.25	3.73	0.28	1.57	0.28	2.08	0.37	2.28
Zr93	0.25	0.51	0.23	0.50	0.16	0.32	0.23	0.96	0.25	0.75	0.17	0.71
Nb93m	1.28	3.88	1.02	3.21	0.60	2.22	1.47	3.56	1.29	2.65	0.81	1.99
Tc99	0.22	0.79	0.22	0.68	0.18	0.51	0.21	0.93	0.23	0.78	0.15	0.75
Ru106	0.91	3.76	1.05	4.51	0.52	4.53	0.93	4.59	1.74	5.23	0.69	4.97
Rh106	0.90	13.84	0.74	13.68	0.54	12.47	0.80	8.38	1.69	8.49	0.63	8.11
Pd107	0.63	1.32	0.49	0.91	0.29	0.78	0.26	0.88	0.23	0.71	0.16	0.66
Cd113m	0.69	1.55	0.60	1.32	0.41	1.00	0.39	1.02	0.28	0.91	0.20	0.57
Sn126	0.38	0.76	0.35	0.69	0.24	0.52	0.23	0.95	0.23	0.69	0.16	0.56
Sb125	0.35	1.31	0.41	1.49	0.17	1.49	0.30	1.23	0.55	1.35	0.26	1.59
Te125m	0.45	2.28	0.41	1.88	0.31	1.31	0.73	2.06	0.49	1.84	0.42	1.50
I129	0.35	0.87	0.33	0.76	0.22	0.50	0.24	0.97	0.26	0.71	0.16	0.63
Cs133	0.23	1.22	0.21	1.03	0.18	0.80	0.27	1.16	0.24	1.07	0.16	0.87
Cs134	0.45	2.42	0.42	2.05	0.19	2.05	0.71	2.24	0.99	2.03	0.48	1.94
Cs135	0.29	0.63	0.20	0.50	0.20	0.47	0.38	0.84	0.22	0.58	0.17	0.44
Cs137	0.22	0.54	0.24	0.55	0.18	0.39	0.23	0.86	0.22	0.64	0.16	0.65
Ba137m	0.33	0.85	0.31	0.66	0.24	0.54	0.28	0.69	0.31	0.91	0.19	0.75
Ce144	0.81	5.28	1.12	5.98	0.48	5.74	1.39	5.93	2.31	6.55	0.80	6.18
Nd148	0.24	0.57	0.27	0.58	0.19	0.42	0.33	0.85	0.22	0.70	0.16	0.62
Pm147	0.43	1.50	0.49	1.64	0.32	1.64	0.50	1.74	0.49	1.69	0.43	2.45
Sm146	2.30	4.37	1.23	2.19	0.52	1.23	1.03	3.94	0.52	1.41	0.37	1.29
Sm147	0.58	3.23	0.56	2.69	0.44	2.17	1.65	3.14	1.43	2.59	1.05	1.83
Sm149	0.62	5.13	0.65	5.56	0.61	5.32	0.68	2.96	0.83	4.20	0.86	3.66
Sm151	0.33	1.69	0.28	1.65	0.31	1.65	0.47	1.25	0.34	1.26	0.28	1.01
Eu154	0.35	1.22	0.25	0.76	0.17	0.75	0.43	1.21	0.39	1.15	0.22	0.84
Eu155	0.37	1.56	0.34	1.05	0.23	1.05	0.46	1.12	0.48	1.20	0.30	0.99
Ho166m	1.73	5.16	0.97	2.71	0.58	1.83	1.37	3.74	0.56	1.63	0.41	1.19
Pb206	3.97	8.87	1.94	7.81	0.69	3.86	9.55	19.55	3.65	7.80	1.39	7.22
Pb207	2.16	5.51	1.34	6.88	0.56	3.18	5.72	14.22	2.87	6.89	1.22	6.41
Pb208	3.61	8.00	1.85	6.90	0.62	3.41	4.62	13.86	3.14	7.01	1.40	5.72
Pb210	0.85	3.42	0.57	3.33	0.49	2.12	2.36	8.88	1.55	7.05	0.90	5.72

	UOX (3-5%)						MOX (5-9%)					
Burnup	25-38		38.1-50		50.1-70		25-38		38.1-50		50.1-70	
Percentile	80	100	80	100	80	100	80	100	80	100	80	100
Bi209	7.08	14.09	2.48	8.18	0.72	4.92	5.60	14.88	3.57	7.83	1.53	6.87
Ac227	0.59	3.26	0.41	2.84	0.46	1.43	2.55	6.75	1.96	4.99	1.08	4.02
Th228	0.88	5.15	0.85	4.85	0.54	3.39	3.04	8.86	2.99	6.85	1.70	5.19
Th229	1.38	3.04	0.83	2.10	0.41	1.34	2.02	7.11	1.18	2.95	0.73	2.61
Th230	1.70	5.48	1.42	5.30	0.63	4.41	1.62	4.37	1.50	3.92	1.16	2.57
Th232	0.27	0.74	0.34	0.77	0.38	0.92	0.45	1.22	0.40	0.82	0.31	2.29
Pa231	0.79	1.30	0.76	1.64	0.62	1.80	0.84	2.48	0.65	2.24	0.49	1.86
U232	0.45	1.28	0.32	1.10	0.33	1.03	1.72	5.95	0.90	3.32	0.54	2.31
U233	1.58	4.52	1.08	3.40	0.93	3.20	0.80	2.62	0.58	2.15	0.44	1.36
U234	1.28	3.64	1.19	4.08	0.59	3.57	0.86	2.47	0.87	1.98	0.78	1.58
U235	0.66	1.25	0.98	3.59	6.11	12.77	0.29	0.74	0.40	1.22	0.58	3.15
U236	0.31	0.61	0.23	0.53	0.23	0.93	0.34	0.74	0.26	0.61	0.22	1.97
U238	0.02	0.07	0.03	0.07	0.02	0.12	0.02	0.05	0.03	0.06	0.03	0.22
Np237	0.38	2.19	0.31	1.71	0.21	1.42	0.28	1.48	0.29	1.39	0.18	1.07
Pu238	1.64	4.39	0.97	2.47	0.35	1.69	0.26	1.06	0.42	1.21	0.41	1.10
Pu239	0.34	1.26	0.22	1.56	0.22	1.56	0.29	1.23	0.57	1.54	0.51	1.46
Pu240	0.30	0.64	0.19	0.54	0.14	0.37	0.19	0.69	0.30	0.72	0.29	0.81
Pu241	0.32	0.86	0.18	0.55	0.13	0.46	0.15	0.49	0.22	0.44	0.17	0.95
Pu242	0.91	3.79	0.63	1.30	0.38	1.05	0.17	0.60	0.28	0.59	0.21	0.47
Pu244	6.71	18.81	1.90	6.72	0.98	3.37	0.61	1.51	0.35	1.13	0.31	0.88
Am241	1.40	7.01	1.20	6.90	0.77	6.39	1.20	2.78	1.34	3.22	1.32	2.38
Am242m	0.89	1.99	0.91	2.59	0.79	2.59	0.51	1.42	0.46	1.38	0.61	1.62
Am243	2.41	7.42	0.77	2.16	0.49	1.50	0.28	0.72	0.24	0.80	0.16	0.92
Cm242	2.27	8.00	1.70	9.50	0.67	9.86	2.39	10.77	3.56	11.05	1.74	10.8
Cm243	2.00	5.72	0.80	2.25	0.38	1.61	0.63	1.44	0.47	1.30	0.34	0.74
Cm244	6.91	20.69	1.75	5.67	0.79	2.84	0.46	1.13	0.29	0.79	0.19	0.67
Cm245	15.52	81.64	3.22	8.14	0.91	3.25	0.86	3.37	0.47	1.23	0.24	0.82
Cm246	58.19	157.22	7.77	23.19	2.31	9.98	5.38	16.77	1.27	4.92	0.62	2.05
Cm247	4.63E2	1.88E3	1.83E1	6.42E1	3.11	17.81	14.21	51.84	2.30	8.81	0.98	2.77
Cm248	2.59E3	1.87E4	3.96E1	3.85E2	4.09	34.02	34.26	99.97	4.33	19.66	1.72	7.24
Cm250	1.91E4	2.01E5	1.56E2	2.19E3	6.41	61.85	67.61	6.28E2	10.12	40.82	3.70	16.2
Bk249	1.19E4	1.39E5	1.24E2	1.41E3	6.24	70.15	60.04	4.77E2	6.83	29.17	3.33	13.5
Cf249	1.91E4	1.49E5	1.07E2	2.23E3	10.26	108.9	89.00	1.40E3	18.64	47.62	3.80	26.2
Cf250	2.71E4	2.94E5	1.58E2	3.38E3	12.19	131.1	98.51	1.89E3	22.65	60.62	4.74	30.9
Cf251	9.41E4	1.34E6	7.03E2	9.06E3	17.41	154.4	203.3	3.08E3	20.88	76.66	6.94	33.6

POLITÉCNICA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es