UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Aproximación hacia la identificación automática de lesión de ictus en TC craneal Documentación Técnica

Presentado por Bárbara Sainz Crespo
en la Universidad de Burgos
3 de julio de 2019
Tutores: Dr. Álvar Arnaiz González
y Dr. José Francisco Díez Pastor

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

La planificación es una parte muy importante de cualquier tipo de proyecto. Es ahí donde se hace una estimación de algunos de los recursos principales que va a suponer la realización del trabajo, en términos de tiempo, dinero, etc.

Para llevar a cabo una buena planificación, se debe hacer un análisis minucioso que contemple todas y cada de las partes que componen el proyecto. Esto ayuda a poder identificar mejor los recursos necesarios para cada parte, a la par que es de gran ayuda en las fases posteriores, ya que una buena organización y agenda de trabajo facilitan un correcto desarrollo del proyecto.

Esta sección del anexo consiste precisamente en detallar todo ese proceso de planificación, que a su vez se puede subdividir según las temáticas y recursos que estemos contemplando. Teniendo esto en cuenta, la planificación se puede desglosar como sigue:

Planificación temporal: elaboración de un calendario o programa en el que se marcan las tareas o partes que componen el proyecto junto con su estimación de tiempo necesario para realizarlas. Se marca, por tanto, una fecha de inicio y fin estimadas para cada tarea o evento. Lo ideal sería anotar también los pesos y requerimientos previos de cada una de las partes.

Estudio de viabilidad: la viabilidad del proyecto viene determinada a su vez por dos factores importantes:

Viabilidad económica: estimación de costes y beneficios de supone la realización del proyecto.

Viabiliadad Legal: análisis del conjunto de leyes que regulan y afectan al proyecto. Cuando se trata de algún tipo de diseño o *software* esto se ve reflejado en forma de licencias de uso, protección de datos, etc.

A.2. Planificación temporal

Para llevar a cabo y al día la planificación temporal de este trabajo, se plantea desde un principio apoyarse en GitHub y ZenHub. Aunque, hay que decir que por la naturaleza del proyecto (individual y educativo), sus circunstancias (distancia y dedicación parcial por trabajo) y la falta de experiencia y costumbre de usar este tipo de herramientas, la realidad es que no se ha seguido al 100 % como se debería.

No obstante, sí se han tratado de seguir ciertas líneas generales:

- Reuniones con cierta frecuencia (1-2 semanas) para comentar y evaluar lo que se ha avanzado desde la última reunión, así como planificar el siguiente periodo temporal hasta la próxima reunión.
- Traducción de dichos periodos temporales planificados entre reuniones a *milestones* o *sprints*, para una estrategia incremental.
- La planificación de cada *sprint* incluía una serie de tareas a realizar, que se traducían en *issues*.
- La estructura principal de cada tarea o *issue* constaba de los siguientes parámetros:
  - Título descriptivo.
  - Breve descripción o desglose cuando corresponda.
  - Asignación de recursos humanos correspondientes (al ser un trabajo individual, en general esto significaba autoasignármelo, pero también los tutores participaban en esta fase).
  - El *milestone* o *sprint* al que pertenecían.
  - Asignación de una estimación de su coste.

Para aclarar este último punto, he de decir que a la hora de estimar el coste se utilizaron los *story points* de ZenHub. Cuando hablamos de coste nos referimos en este caso a coste temporal, así que básicamente se trataba de estimar el tiempo aproximado que llevaría cada tarea.

En este sentido, se utilizaron los distintos valores de los *story points* como si fueran horas de trabajo que se calculó consumiría cada tarea o *issue*. Exactamente así, es decir, un *story point* de 4 significa para nuestro proyecto que la tarea se estima que llevará 4 horas y, cuando se indica 1, se engloba una posible tarea de 30 minutos, ya que se entiende este valor como cota máxima.

Por otro lado, en el caso particular del desarrollo de este trabajo, se produjeron algunas interrupciones debido principalmente a circunstancias ajenas a nuestro alcance, como son problemas de salud, pero también alguno técnico, sobretodo durante los primeros meses. Hay que tener en cuenta que, al tratarse de un proyecto individual, el hecho de que el principal recurso humano y único contribuidor cause una baja temporal, afecta al 100 % de la planificación, organización y desarrollo del proyecto. En general, esto se tradujo en una pausa o en la prolongación temporal del *sprint* en curso.

Una vez explicado todo esto y antes de proceder a detallar el contenido de los *sprints* que han constituido la planificación del presente proyecto, he de decir que previamente a la reunión que marca el comienzo del primer *sprint* (y con ello del proyecto), ya habían tenido lugar encuentros y conversaciones por asuntos referentes a este proyecto. En estas reuniones, se trató en persona con los tutores, y en un par de ocasiones también con la neuróloga con la que colaboramos vía videoconferencia, para sentar las bases y dar forma al proyecto acordando el plan a seguir sobre ciertos temas:

1. Comunicar a los tutores la idea que tenía en mente, junto con la Dra. María Hernández Pérez del Hospital Germans Trias de Barcelona, en el que trabaja como neuróloga, con dedicación a partes iguales clínica e investigadora. Se plantea la posibilidad de hacer un acuerdo de colaboración institucional con la Universidad de Burgos, para abrir una rama relacionada con su investigación, pero hacia las aplicaciones de Ingeniería Informática que podrían ayudar en la misma, en concreto el procesamiento de imágenes 3D en el ámbito de la Inteligencia Artificial. Dicha colaboración e investigación comenzaría con este proyecto, aunque sabíamos que sólo podría constituir una parte de nuestra idea en sí.

2. Objetivos planteables a nivel del TFG, ya que la idea general del proyecto iba más allá y se correspondía más con un trabajo de años, como una tésis doctoral. Acordamos centrarnos en esta parte, relacionada con la segmentación de imágenes (TC craneales), para tratar de aproximarse a identificar automáticamente lesiones de ictus.

3. Entonces, se produce una reunión con todas las partes colaboradoras, para sentar las bases y aclarar conceptos unos a otros, tales como con qué cantidad de imágenes podríamos contar, cómo son exactamente, tiempos aproximados para poder tenerlas y empezar a investigar con ellas o qué podrían esperar en nuestra opinión.

4. Se me plantea la posibilidad de ir a trabajar al extranjero, por lo que nos reunimos de nuevo para ver si podrían ser ambas cosas compatibles o cómo cambiaría esto en el planteamiento que estábamos aún asentando.

5. Finalmente, decidimos seguir adelante y replantearlo ya para el segundo cuatrimestre adaptándolo ligeramente a las nuevas circunstancias, con reuniones aproximadamente cada 2 semanas por videoconferencia, acceso remoto, etc.

En conclusión, seguiremos en contacto y comenzando a trabajar en el acuerdo de colaboración, de momento, para ya más adelante, una vez reubicada y asentada en mi nuevo destino, comenzar el proyecto en sí.

A partir de aquí, comienzan los *sprints*, con sus *issues* y se pueden analizar con *Burndown* como sigue a continuación.

Sprint 1 (23/01/2019 - 30/01/2019)

Este primer *sprint* comienza con una "reunión inicial" en la que se acuerda empezar con la parte de investigación del estado del arte en temas relacionados con la segmentación de técnicas de neuroimagen 3D, o en su defecto de imágenes tridimensionales Biomédicas en general.

Además, se crea un repositorio en GitHub para gestionar el desarrollo del proyecto y se plantea la idea de explorar las posibilidades de ZenHub para apoyar en cuanto a la planificación del mismo, una herramienta que no conocía. Lo cierto es que personalmente, tampoco estaba muy familiarizada ni acostumbrada a trabajar con GitHub, así que tocaba refrescar la memoria y revisar un poco su manejo y opciones. Teniendo esto en cuenta, junto con la idea de empezar poco a poco (para acostumbrarse a combinar el proyecto

con la vida laboral) y aumentar progresivamente la carga de trabajo, se explica que las *issues* son bastante pobres inicialmente.

Issues del *Sprint01*:

- Leer un artículo.
- Leer otro artículo y e ir encontrando otros relacionados a partir de ahí.

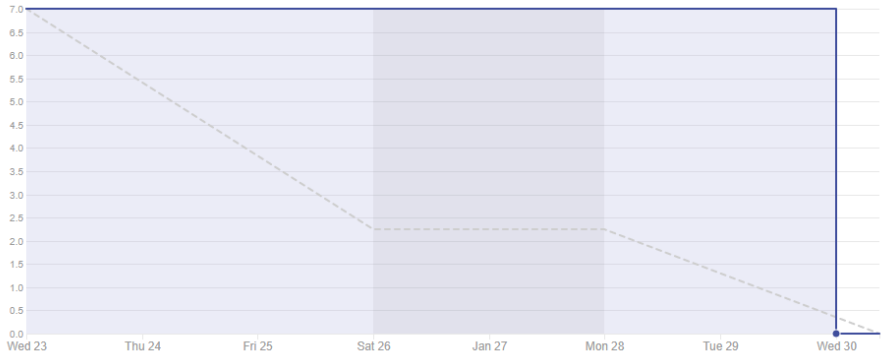

Figura A.1: Sprint 01

A cada artículo se le estimaron aproximadamente 3-4 horas, aunque el tiempo dedicado fue algo mayor, sobretodo a la hora de sacar información de ellos para dar con otros artículos, los cuales no añadimos como *issues* por despiste. Por otro lado, aunque no se añadió, anotamos descargar y revisar y probar el VPN FortiClient, para conectarse con la IP de la Universidad de Burgos. Esto último era necesario para conectar remotamente con la máquina de la universidad, cuyas GPU's ejecutarían el código del proyecto.

Sprint 2 (28/03/2019 - 10/04/2019)

Terminar de leer artículos y añadir y leer los que se fueorn sacando. También se propone organizar la información de todos listada en una hoja de cálculo. También se acuerda descargar y trastear con LaTeX para valorarlo como opción para la documentación del TFG.

La clave ahora es tratar de encontrar encontrar algún *software*, códifo abierto o conjunto de herramientas en lo que basar nuestro proyecto.

Issues del *Sprint02*:

- Leer sobre el *dataset* ATLAS.
- Instalar LaTeX y configurar el editor *TeXstudio*.
- Organizar info de los artículos leídos.
- Leer artículo sobre segmentación automática de lesión del ictus con CNN [1].
- Leer artículo sobre análisis de métodos de imágen médica en ictus isquémico [3].
- Leer artículo sobre ISLES 2018 [4].

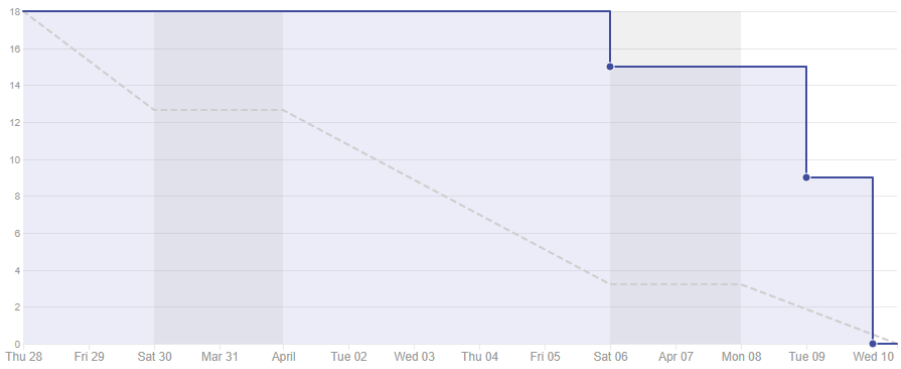

Figura A.2: Sprint 02

Unas 18 horas de trabajo estimadas en total, aunque en esta fase solía tardar algo más de lo estimado.

Sprint 3 (11/04/2019 - 26/04/2019)

Se trata de lograr acceder al servidor en remoto, crear un entorno virtual en Python, leer algunos artículos más, instalar NiftyNet y probar el modelos de los TC abdominales. También se acuerda echar un ojo a repositorios de imágenes y a cómo han de ser para la entrada al sistema.

Issues del *Sprint03*:

- Leer sobre el aprendizaje profundo en imágenes médicas.
- Revisar bien la fuente de código y marco de trabajo de NiftyNet.

- Aprender y valorar V-Net como opción de CNN.
- Búsqueda entre marcos de trabajo y código en que basarnos, referente a imagen Biomédica.
- Acceder al servidor y manejarse sin problemas.
- Aprender sobre entornos virtuales, para crear uno propio.

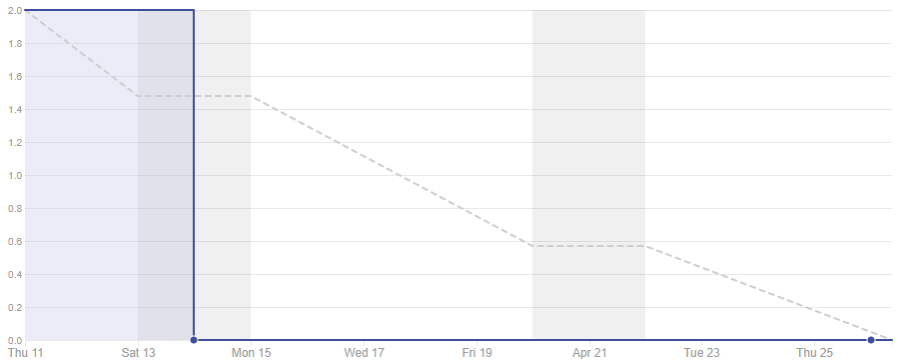

Figura A.3: Sprint 03

Por desgracia y por error, varios de los *issues* de este *sprint* no se estimaron. Luego la información que aporta este *sprint* no es relevante como tal.

Sprint 4 (27/04/2019 - 15/05/2019)

El tratamiento de imágenes cobra importancia y ya se nota la subida prevista de la carga de trabajo.

Issues del *Sprint04*:

- Leer y cargar el formato .nii
- Pequeño entrenamiento de prueba con DeepMedic.
- Revisar modelo de entrenamiento en NiftyNet, para TC abdominal.
- Comprobar formatos de TCs
- Formatos .dcm y .nii

- Abrir y revisar imágenes de nuestra muestra de datos.
- Instalar MRIcron y aprender a usar dcm2nii para convertir archivos a NIfTI.
- Consultar el modelo DeepMedic

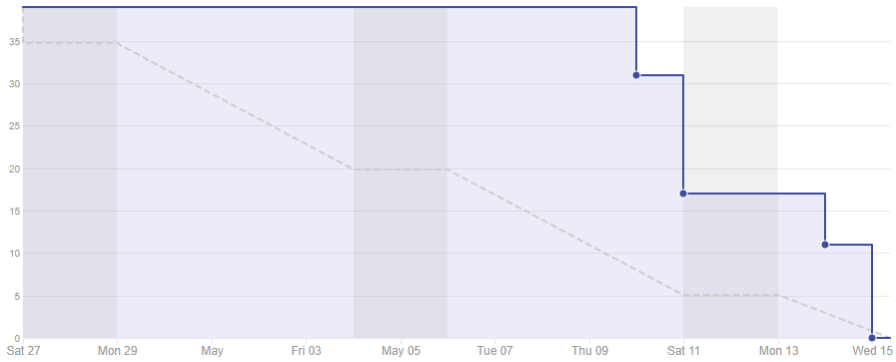

Figura A.4: Sprint 04

Planteadas 39 horas, aunque atascos imprevistos con las instalaciones y pruebas aumentaron el tiempo real invertido.

Sprint 5 (16/05/2019 - 24/05/2019)

Los objetivos son abrir, entender y visualizar archivos en formato nifti (.nii / .nii.gz), así como averiguar cómo etiquetar ROIs en nuestras imágenes .nii para tener un input con el que entrenar. Esto respecto al procesado de imágenes, para poder empezar a hacer pruebas con nuestro tipo de datos que nos interesan. Paralelamente, revisar y descargar datos de TCs con ictus etiquetados de ISLES 2018, por si es una opción más rápida para comenzar a probar.

Por otro lado, comenzar la documentación del TFG, y también comentar y hacer *commits* de lo hecho hasta ahora en GitHub.

Issues del *Sprint05*:

- Revisar la documentación de proyectos anteriores, para hacerse una idea de cómo es la documentación.
- Descargar datos de ISLES 2018.

- Hacer commits, comentar y subir código utilizado hasta ahora, empezando a rellenar el repositorio de GitHub.
- Visualizar archivos nifti
- Preparar y etiquetar TCs para entrenamiento.

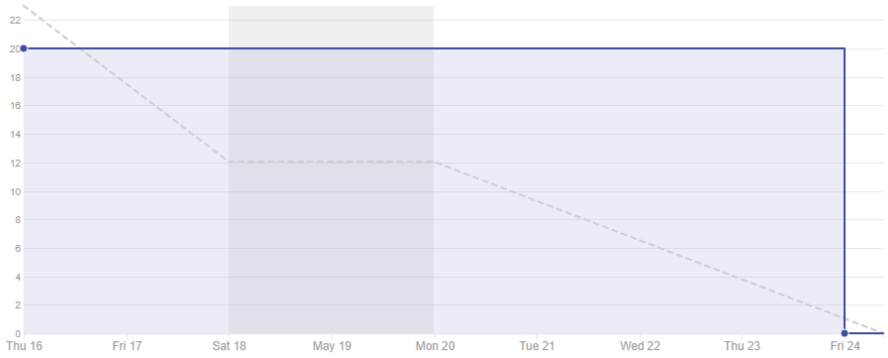

Figura A.5: Sprint 05

Se llegan a programar "solo "23 horas (bajada respecto a la anterior etapa), sin embargo esto es porque fueron muchas más en tiempo real, no dando pie a seguir creando más *issues*. El preparado y etiquetado, estimado en 8 horas, se tradujo en varios días de investigar, hacer pruebas, contactos con los expertos, etc. En la memoria se explican las causas que finalmente se identificaron y cómo se solucionaron.

Sprint 6 (25/05/2019 - 11/06/2019)

En mitad de este *sprint*, se llevó a cabo una reunión en persona, la primera desde el comienzo de desarrollo del proyecto, el día 03/06/19 en la Universidad de Burgos, aprovechando mi viaje de vuelta para unos días. No obstante, se decidió mantener el mismo *sprint* prolongado puesto que las tareas no cambiaban, prácticamente durante la semana posterior.

Issues del *Sprint06*:

- Ejecutar notebook en Gamma
- Entrenar una nueva red propia (estudio de cómo hacerlo y pruebas con un par de imágenes sólo para comprobar que consigamos que funcione sin errores).

- Revisar, comentar y entender NiftyNet con TC abdominales.
- Documentar sobre las técnicas y herramientas generales.
- FTP, envío TCs preparados y visualización de ficheros de salida tras la segmentación de TC abdominal.
- Probar cómo (si) funciona con nuestros propios TCs etiquetados en niftynet.

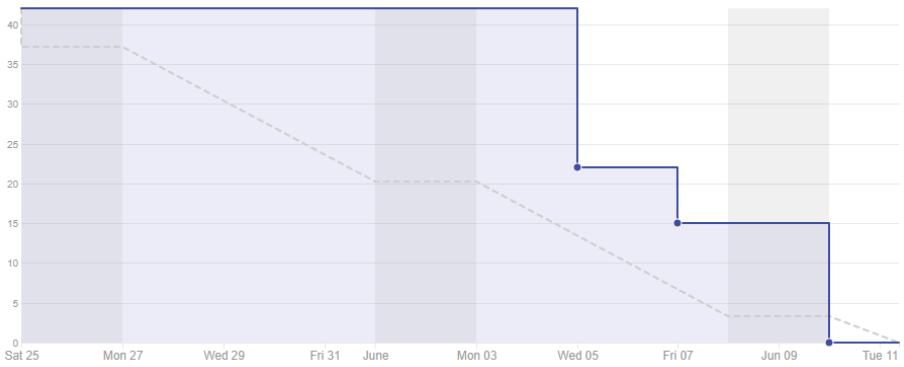

Figura A.6: Sprint 06

Son 42 las horas estimadas, aunque de nuevo un par de ellas se alargan, al costar interpretar algunos errores o encontrar información del entrenamiento propio, por ejemplo, donde la documentación de NiftyNet no era tan clara y específica, ni aportaba ejemplos, solo de segmentación con los abdominales.

Sprint 7 (12/06/2019 - 18/06/2019)

El final del *sprint* anterior y comienzo de este viene marcado por un inconveniente imprevisto que supone una alteración en la planificación que a priori se tenía en mente para las semanas restantes. Esto es, un problema con el convenio o acuerdo de colaboración, cuyo error implica que todo el procesamiento de imágenes hasta el momento, el cual ya se ha visto supone mucho tiempo, va a haber que repetirlo con nuevas imágenes TC que iremos recibiendo en adelante, puesto que los datos hasta ahora no pueden utilizarse.

Issues del *Sprint07*:

- Revisar las correcciones y comentarios hechos en la memoria hasta le momento.
- Modificar notebooks.
- Completar técnicas y herramientas en la documentación
- Documentar conceptos teóricos.
- Corregir documentación, basada en las correcciones y comentarios.
- Rellenar/redactar nuevo convenio de colaboración.
- Modificar ficheros del repositorio y su estructura
- Modelo para segmentación con TCs craneales propios
- Revisión y preparación (formato y etiquetado) de nuevos TCs (según nuevo convenio).
- Primera prueba de entrenamiento propio
- Revisar los últimos 2 notebooks y el contenido del directorio extensions
- Finalizar mini entrenamiento y segmentación sin errores

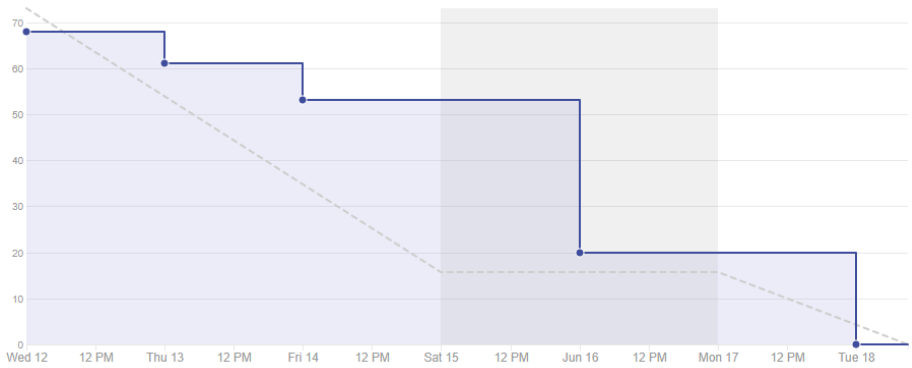

Figura A.7: Sprint 07

Se da el salto a 73 horas, en *sprints* de menos días, para culminar el aumento de carga de trabajo hacia el final, aprovechando que este mes se pausa la vida laboral, para dedicarme de lleno al proyecto. Además, esta vez sí sabemos estimar mejor el tiempo que supone el tratamiento y preparación de imágenes.

Sprint 8 (19/06/2019 - 24/06/2019)

En estos 6 días, seguimos procesando o preparando nuevas imágenes TCs (formato, dimensiones y etiquetado), para prepararlas para un entrenamiento que llegue a buen puerto.

Con ello, se pretende por fin llegar a hacer alguna prueba de entrenamiento con más de dos TCs y un número de iteraciones más importante que pueda llegar a aportar algún resultado que apunte maneras. Para comprobar esto, se pretende hacer pruebas de segmentación con los resultados de los entrenamientos.

Por otra parte, hay que seguir avanzando en la documentación.

Issues del *Sprint08*:

- Prueba de entrenamiento con 2 TC de distinto número de cortes usando V-Net.
- Prueba de entrenamiento con 2 TC de distinto número de cortes usando MyNet.
- Procesado de nuevos TC para prepararlos para entrenamiento
- Recopilar y dejar listos 10-20 TCs para probar a entrenar el sistema con ellos.
- Entrenamiento con 10-20 TCs preparados.
- Terminar de documentar conceptos teóricos (IA y CNN).
- Documentación: DeepMedic, NiftyNet y V-Net
- Probar segmentación con el resultado del entrenamiento de 10-20 TCs
- Comentar código subido al repositorio.

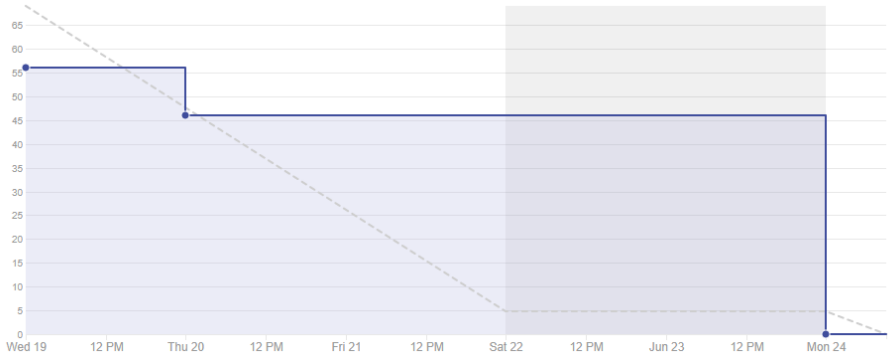

Figura A.8: Sprint 08

Esta vez se llega a 69 horas de nuevo mucho mejor estimadas con respecto a la realidad, contando con un día menos que el *sprint* previo.

Sprint 9 (25/06/2019 - 29/06/2019)

Se trata de probar a evaluar matemática y analíticamente los resultados obtenidos al segmentar. También realizar un último entrenamiento con más imágenes (cerca de 50 se espera) en cuanto termine el etiquetado de las últimas.

Por otro lado, seguir con la documentación, con el objetivo de dejar prácticamente terminada la memoria con los datos obtenidos hasta le momento al menos, a falta de anexos y posibles pequeñas modificaciones para los últimos días.

Issues del *Sprint09*:

- Terminar de etiquetar los últimos TCs.
- Ubicar datos para entrenamiento final.
- Entrenamiento final con 40-50 TCs y 3000 iteraciones.
- Documentar etiquetado de TCs
- Documentación de introducción y objetivos del proyecto.

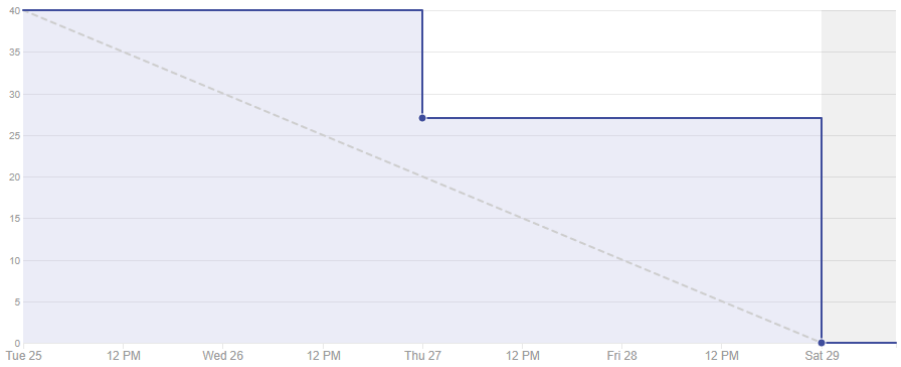

Figura A.9: Sprint 09

Hay dos alteraciones importantes en este caso: un problema técnico con GitHub imposibilita el *push* de nuevos ficheros y modificaciones al repositorio, por un lado, y por otro se termina de descubrir con los etiquetados que numerosas imágenes TC del convenio nuevo, en realidad no presentan lesión de ictus.

Entre una cosa y otra, se despista un poco el correcto uso de GitHub/ZenHub, ya que no se incluye *issue* de las tareas que se llevan a cabo para tratar de reaccionar ante este segundo imprevisto por la espontaneidad de las mismas, lo ajustado que está el tiempo y la imposibilidad de actualizar el repositorio que de alguna manera implica acordarse menos y dejarlo un poco de lado.

Con todo, quedan "sólo" estimadas 40 horas para estos 5 días. La estimación del entrenamiento fue esta vez algo pesimista, por lo que se compensó algo con el tiempo real invertido en total, si le añades el procesamiento y revisión de imágenes para buscar hacer otro entrenamiento más, diferente al planteado para este *sprint*. Además, todo el proceso para este último entrenamiento se comienza en este sprint, pero dado que se termina en el siguiente, su *issue* correspondiente (estimado en 48 horas, aunque se pudo rebajar un poco) realmente pertenece parcialmente al tiempo de trabajo invertido en éste. Esto implica reflejado en los datos una infraestimación de coste temporal para este *sprint09*, que se verá compensado con la sobreestimación del siguiente y final.

Sprint 10 (30/06/2019 - 3/07/2019)

En los últimos días se requiere terminar la documentación de aspectos relevantes de la memoria y los anexos, y revisión y corrección de la misma. Asimismo, se han de evaluar los resultados obtenidos y compararlos.

Si diera tiempo, también se acuerda la realización de un último entrenamiento, cuya preparación y puesta en marcha ya fue comenzada en el *sprint* anterior, como acabamos de comentar.

Issues del *Sprint10_Final*:

- Probar evaluación de las segmentaciones realizadas.
- Documentación sobre V-Net.
- Probar un pequeño entrenamiento vnet (no densa).
- Último entrenamiento.
- Finalizar documentación de la memoria
- Ajustes en conceptos teóricos de la documentación.
- Evaluación final.
- Terminar anexos. (Aún abierto en el momento de escribir esto, por lógica).

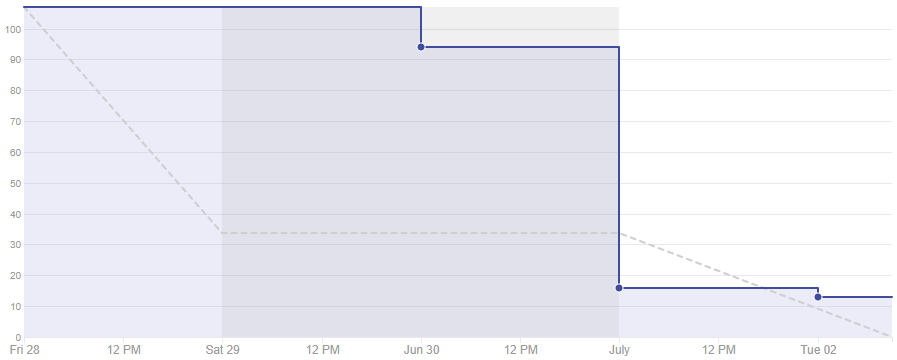

Figura A.10: Sprint 10_Final

Finalmente, la estimación de este último *sprint* se queda en 107 horas, para los últimos 5 días y medio, aunque no todo es tiempo real de estos

días como se ha explicado. Por su parte el tiempo de la estimación para el entrenamiento final se pasa, es decir, fue de unas 8 horas menos, aunque la de terminar los anexos se ha quedado corta, como prueba que ya se está haciendo fuera del tiempo planificado en este momento.

En conclusión, entre los imprevistos y los despistes o errores, se hace autocrítica en cuanto a que el proceso de planificación temporal no ha sido el más adecuado en algunas fases del proyecto. Bien es cierto, que aún así, las horas estimadas totales no han quedado demasiado lejos de las reales en más de un *sprint*, en otros algo por encima y luego hay dos excepciones claras donde sí fue muy superior por no anotar las *issues* correctamente. Esto es, en el *sprint03* donde prácticamente no se registraron estimaciones, y el *sprint01* donde no se anotaron directamente las tareas según se fueron encontrando y trabajando más artículos.

El total de horas estimadas para el desarrollo del proyecto (sin contar las reuniones previas, ni las llevadas a cabo entre cada *sprint*) ha quedado en 420 horas. Mientras que el real ha estado cerca de las 480 horas registradas. De nuevo, sin contar reuniones ni la investigación y aprendizaje personal inicial en cuanto a comprender el ámbito en el que nos movíamos, en general.

A.3. Estudio de viabilidad

En este apartado se procede a estudiar la viabilidad del proyecto que nos ocupa, tanto a nivel de costes o económico, como legalmente hablando.

Viabilidad económica

Para el estudio de la viabilidad económica, se van a aproximar los costes que supondría el desarrollo de este proyecto, como si lo llevara a cabo una empresa. En este sentido, hay que tener en cuenta los costes por recursos de personal, así como materiales.

Coste del personal

Tal como se ha mencionado antes, la realización de este proyecto al completo ha necesitado de cerca de 480 horas de desarrollo, sin contar con las 13 reuniones con los supervisores ni las 8 con la experta perteneciente al hospital con que se colaboraba. Estas reuniones podrían promediarse en un tiempo de hora y media cada una, aproximadamente. Esto supone un total de casi 3 meses y medio de dedicación completa. Contando con un tiempo

previo para el planteamiento de la idea y una formación mínima, podemos aproximarlo a cuatrode empleo a tiempo completo.

Suponiendo que este trabajador obtenga un salario bruto de 1500 euros (algo más de 1200 euros netos), habría que calcular los gastos de Seguridad Social por parte de la empresa y de dicho trabajador:

Empresa :

- Contingencias comunes: 23,6 %
- Desempleo: 5,5 %
- Formación profesional: 0,6 %
- Fondo de Garantía Salarial: 0,2 %

En total, supone un $29,9\,\%$ por parte de la empresa.

Empleado :

- Contingencias comunes: 4,7 %
- Desempleo: 1,55 %
- Formación profesional: 0,1 %

El total en cuanto al trabajador supone un $6,35\,\%$. Lo cual afecta al sueldo neto, que quedaría en 1224,75 euros tras descontar esto (95,25 euros) y un IRPF del 12 %, claro que esto podría variar según las circunstancias personales del empleado a la hora de declarar a Hacienda.

Estos datos se han extraído de las bases de cotización publicadas en la página web oficial de la Seguridad Social.

Dado que lo calculamos para cuatro meses, el total del salario bruto serían 6000 euros, luego los gastos totales por parte de la empresa ascenderían a 1794 euros.

Por otra parte, se ha contado con dos supervisores (tutores del proyecto), durante el mismo periodo de cuatro meses. No he encontrado las cifras concretas para el año vigente específicamente publicadas en la página web de la Universidad de Burgos, pero basándome en las que sí he encontrado un poco más antiguas haré la siguiente aproximación:

El sueldo mensual como doctor sería de unos 2350 euros, es decir, 28200 anuales. Como profesional permanente imparte un total de 32 créditos, luego

cada crédito supone 881,25 euros. Aplicando 0,5 ECT por la supervisión de este proyecto, el sueldo de cada tutor por el mismo sería de 440,625 euros cada uno. Entonces, el coste que supondría por ambos supervisores ascendería aproximadamente a unos $881,25$ euros.

Por tanto, el coste total de personal, incluyendo sueldo y gastos de SS del trabajador y salario de los supervisores ascendería a $8675,25$ euros.

Coste material

Todo el código de las librerías utilizadas como base del proyecto era libre, es decir, no supone ningún gasto de *software*.

Mientras que a nivel de *hardware* el mínimo necesario sería un ordenador portátil para el trabajador, cuyo valor podemos estimar en 800 euros. Este tipo de material se amortiza en 5 años, luego en un periodo de cuatro meses el coste que supondría sería de:

$$(800euros/(5años * 12meses/año)) * 4meses = 53,33euros$$

El coste de la máquina donde con la que realmente se llevan a cabo los procesos es muy difícil de calcular, por dos motivos principales:

- A priori, desconozco el valor concreto de este tipo de software, ya que no dispongo de todas sus especificaciones.
- Aún si pudiera hacer una estimación del valor del procesador, se entiende que la empresa no lo compraría y utilizaría solo para este proyecto en concreto, con lo cual una amortización de 4 meses sería más que excesivo por las horas que realmente se use en este proyecto, ya que eso es lo que le supone a la empresa teóricamente tenga los proyectos que tenga en marcha durante ese periodo.

Por tanto, asumimos que se trata de una empresa que ya tiene tal procesador en uso, como ha sido el caso de la Universidad de Burgos cediendo su uso para este proyecto y no añadiremos coste extraordinario por ello, ya que en las cuentas de la empresa igualmente tendría que amortizar este periodo temporal.

Finalmente, consideraremos algunos otros gastos generales por el alquiler de un espacio de trabajo durante los cuatro meses y los gastos de electricidad, Internet, impresora, etc. que esto conllevaría. Estimamos la suma de todo esto en un total de unos 800 euros.

Así, la suma de los gastos materiales ascendería a unos $853,33$ euros.

Finalmente, el cómputo global de los gastos estimados del proyecto sería:

$$8675,25(personal) + 853,33(material) = 9528,58euros$$

Viabilidad legal

Teniendo en cuenta las herramientas utilizadas, las licencias legales de uso serían como siguen:

Deepmedic: BSD 3-Clause License.

NiftyNet: Apache License, Version 2.0 [1]

TensorFlow: Apache License, Version 2.0

MRIcron: Chris Rorden's MRIcron

MicroDicom: Copyright (c) 2007-2016 MicroDicom[2]

Respecto a la legalidad en cuanto a los datos utilizados, se realizó un convenio de colaboración específico propio entre las entidades implicadas (Universidad de Burgos y Hospital Germans Trias i Pujol de Barcelona), para el uso legal de las mismas en este proyecto. Éste fue firmado por las partes conforme a la legalidad vigente y aprobado por el Comité de Ética del hospital en cuestión.

[1] Copyright 2018 the NiftyNet Consortium.
[2] Especificaciones en http://www.microdicom.com/eula.html

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este apartado se refiere a los requisitos globales del proyecto presentado, así como a los funcionales implementados.

B.2. Objetivos generales

Tal como se definía en la memoria, el principal objetivo del proyecto es el diseño de un sistema inteligente experimental, entrenado mediante aprendizaje supervisado, que permita una aproximación a la identificación de la zona del cerebro afectada o lesionada a causa de un ictus, así como su delimitación en un TC craneal tras 24h de la incidencia.

Para ello, los objetivos concretos en cuanto a las herramientas usadas se corresponden con las librerías de *NiftyNet* principalmente, basadas a su vez en las de *TensorFlow*.

B.3. Catalogo de requisitos

Al ser un trabajo de aproximación experimental, no se plantean requisitos específicos para el término de este proyecto, ya que no se está diseñando ningún software o aplicación, si no que se ha investigado si potencialmente concluimos que podría llegarse a tales.

Se podría decir, por tanto, que los requisitos planteados en este caso se resumirían analogamente a un *software* de la siguiente forma:

RF 1 Entrenar un sistema de aprendizaje supervisado para la segmentación de lesión de ictus en TC craneal.

- **RF 1.1** Entrada de datos de TC craneales y máscaras etiquetadas por un experto (ambos .nii).
- **RF 1.2** Procesamiento de datos usando redes neuronales convolucionales.
- **RF 1.3** Devuelve los pesos del modelo entrenado (model.ckpt).

RF 2 Segmentación de lesión de ictus en TC craneal.

- **RF 2.1** Entrada de datos de TC craneales (.nii).
- **RF 2.2** Procesamiento de datos usando los pesos del modelo entrenado.
- **RF 2.3** Devuelve una máscara binaria etiquetando una zona que predice como lesión del ictus (formato NIfTI)).

RF 3 Evaluación de datos segmentados.

- **RF 3.1** Entrada de datos de máscaras con la etiqueta de la lesión (.nii).
- **RF 2.2** Procesamiento de datos usando los pesos del modelo entrenado.
- **RF 2.3** Devuelve una máscara binaria etiquetando una zona que predice como lesión del ictus (formato NIfTI)).

B.4. Especificación de requisitos

Se listan algunos requisitos para el uso experimental del sistema NiftyNet, referidos al catálogo anterior

Caso de uso 1 Entrenamiento del modelo.

Requisitos asociados RF 1, RF 1.1, RF 1.2, RF 1.3

Precondiciones :

1. Entrada de pares de ficheros .nii con ID común en su nombre.
2. Resto del nombre TC o mask, respectivamente. (se puede configurar para cambiar estas palabras clave)

3. Fichero de configuración config.ini con parámetros listados en la documentación de NiftyNet [1]

Acciones :

1. Ejecución del comando de entrenamiento.
2. Listado de configuraciones.
3. Guardado de pesos del modelo

Postcondición : ficheros model.ckpt por cada n iteraciones configuradas para guardar

Excepciones :

1. Datos de entrada incorrectos.
2. Configuración incorrecta.

Caso de uso 2 Segmentación de TC craneal.

Requisitos asociados RF 2, RF 2.1, RF 2.2, RF 2.3

Precondiciones :

1. Entrada ficheros .nii que contentan TC en su nombre (se puede configurar para cambiar esta palabras clave)
2. Entrada de ficheros model.ckpt resultados del entrenamiento
3. Fichero de configuración config.ini con parámetros listados en la documentación de NiftyNet [2]

Acciones :

1. Ejecución del comando de inferencia.
2. Listado de configuraciones.
3. Guardado de máscara con la segmentación.

Postcondición : ficheros .nii.gz por cada par de ficheros de entrada con el mismo ID.

Excepciones :

1. Datos de entrada incorrectos.
2. Configuración incorrecta.

Caso de uso 3 Evaluación de la segmentación.

Requisitos asociados RF 3, RF 3.1, RF 3.2, RF 3.3

[1] https://niftynet.readthedocs.io/en/dev/config_spec.html
[2] https://niftynet.readthedocs.io/en/dev/config_spec.html

Precondiciones :

1. Entrada de pares de ficheros .nii con ID común en su nombre.
2. Resto del nombre _niftynet_out o mask, respectivamente. (se puede configurar para cambiar estas palabras clave)
3. Fichero de configuración config.ini con parámetros listados en la documentación de NiftyNet [3]

Acciones :

1. Ejecución del comando de evaluación.
2. Listado de configuraciones.
3. Guardado de ficheros con métricas de la evaluación.

Postcondición : par de ficheros label.csv y subject_ID_label.csv por cada entrenamiento evaluado (el primero de una línea con las medias y el segundo de una línea dedicada a cada sujeto o par de ficheros con el mismo ID introducido)

Excepciones :

1. Datos de entrada incorrectos.
2. Configuración incorrecta.

[3] https://niftynet.readthedocs.io/en/dev/config_spec.html

Apéndice C

Especificación de diseño

C.1. Introducción

En este proyecto no se ha diseñado ninguna aplicación o software como tal, sino que ha sido de investigación, experimental. Por ello, simplemente procedo a listar lo más parecido análogamente y hago referencia a la memoria, donde ya se explica el diseño teórico que se utiliza.

C.2. Diseño de datos

Respecto a los datos de entrada de cada procedimiento, ya se han explicado sus formatos y cómo procesar las imágenes tanto en los requerimientos como en la memoria.

En cuanto a las redes neuronales utilizadas, hago referencia a su diseño general en los apartados de Redes Neuronales Artificiales y, más en concreto, en el de Redes Neuronales Convolucionales, dentro de la sección de Conceptos teóricos de la memoria. Además, específicamente el diseño utilizado es el de V-Net y V-Net densas, ambas detalladas en la sección de Técnicas y Herramientas de la memoria. Como resultado final se utiliza una arquitectura como la mostrada a continuación:

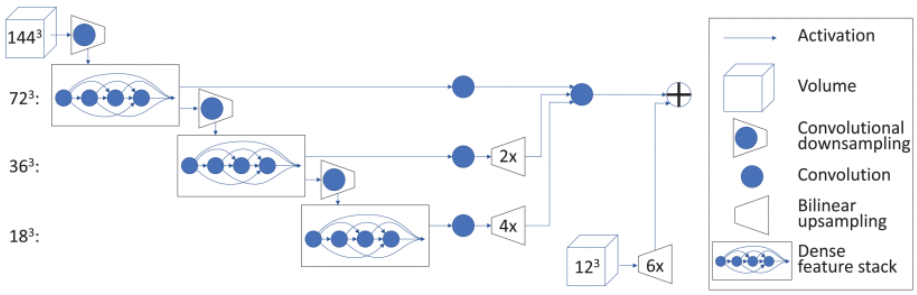

Figura C.1: Representación de la arquitectura de red DenseVNet utilizada. [2]

Aparte de esa clase, explicada también en la sección de aspectos relevantes del desarrollo del proyecto de la memoria, también se probó con mi propia clase *MyNet*, la cual hereda de *BaseNet* que también sigue la misma arquitectura de CNN[1] volumétrica y densa. Simplemente se trataba de probar con menos parámetros, para simplificar en las primeras pruebas.

C.3. Diseño procedimental

Nuevamente los procedimientos para pruebas experimentales están explicados para el desarrollador. En cuanto a su uso, aunque no haya una aplicación como tal, sino solo una aproximación a las técnicas que podrían usarse como base para diseñarla, se supone que el procedimiento a seguir consistiría en la segmentación, ya que los entrenamientos estarían hechos. Al final, son los pesos logrados tras mucha experimentación con los entrenamientos, los que determinarían si se puede llegar a una aproximación lo suficientemente buena para crear con ellos un dispositivo de segmentación automática, los cuales no lo han sido con los recursos de este proyecto, como era de esperar.

En el supuesto caso de lograrlo, el diseño procedimental sería de tal maniera que el usuario introduciría como entrada imágenes de TC, simplemente con ubicarlas en una cierta ruta. Una vez hecho esto habría que diseñar una interfaz para que el usuario simplemente lo ejecutara, mientras que los pesos (el otro dato de entrada para la segmentación serían internos) y podría introducir el nombre de sus archivos para que el sistema se adecuara, sin tener que entrar a cambiarlo manualmente en el fichero config.ini.

[1] Red Neuronal Convolucional

Si, con el tiempo, los desarrolladores fueran mejorando la precisión del entrenamiento con nuevas imágenes y más iteraciones, cada vez que la mejora fuera lo suficientemente importante se podría ofrecer una actualización o un nuevo modelo, que internamente pasara a utilizar los nuevos pesos (model.ckpt) conseguidos.

Entonces, la segmentación devolvería una máscara, para la que podría desarrollarse una arquitectura que directamente la añadiera como *overlay* o capa superpuesta, al TC introducido, que se puede perfectamente estar mostrando por pantalla utilizando el visor que tenga el hospital para ello, integrándolo de esa manera para que el resultado de la segmentación sea directamente visual a los pocos segundos.

C.4. Diseño arquitectónico

Nuevamente, nos referimos de momento a las librerías de NiftyNet, que usa a su vez las de TensorFlow. Aparte de eso, el diseño de los distintos entrenamientos lo llevamos a cabo usando distintas configuraciones y redes, para experimentar, pero no llegamos a cambiar la arquitectura en esta primera aproximación a este campo.

Bien es cierto que se contempla como una línea futura, modificar la función de pérdida de la arquitectura para tratar de aprender de un modo más específico en cuanto a la información de cada pixel y los de alrededor, pero no de sus coordenadas o localización. Ya que esto no tiene relación con la lesión, más allá que la meramente probabilística.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Aquí se explica la estructura de directorios que sigue la entrega del presente proyecto y los pasos necesarios para ejecutar los distintos procedimientos que componen este proyecto. Se trata finalmente de mostrar los pasos a seguir para la instalación de todos los componentes necesarios para hacer las pruebas y seguir desarrollando en función de ellas y los recursos que se tengan mejoras de aproximación y, con ello, del proyecto global.

D.2. Estructura de directorios

Partiendo del directorio raíz y la instalación de *NiftyNet*NiftyNet en el mismo, tendremos un primer directorio padre (niftynet) dentro del cual se desarrolla la estructura de directorios especificada a continuación:

NiftyNet

- /bin
- /include
    - /include/python3.6m/... (102 ficheros *.h* de NiftyNet)
- /lib

    - /lib/python3.6/... (cientos de archivos anidados pertenecientes a las librerías de NiftyNet)

- **/data**
    - **/data/dense_vnet_TC_craneal/** (aquí se introducen los datos de entrada)

- **/extensions**
    - **/extensions/dense_vnet_TC_craneal/___init___.py** (fichero en blanco para comenzar)
    - **/extensions/dense_vnet_TC_craneal/CNN_clase_Red.py** (fichero con la especificación de la clase de red que se quiera utilizar, de ser una diferente a las existentes)
    - **/extensions/dense_vnet_TC_craneal/config.ini** (fichero de configuración inicial con todos los parámetros para usar en el proceso)
    - **/extensions/dense_vnet_TC_craneal/dice_hinge.py** (fichero con la función de pérdida para el entrenamiento)

- **/models**
    - **/models/dense_vnet_TC_craneal/** (directorios y ficheros de salida de los procesos)
        - **.../evaluation_test/** (directorios y ficheros de salida de la evaluación)
        - **.../models/** (directorios y ficheros de salida del entrenamiento)
        - **.../segmentation_output/** (directorios y ficheros de salida de la inferencia)

D.3. Manual del programador

Los parámetros de configuración disponibles también se especifican en la sección de aspectos relevantes del proyecto de la memoria, además de en el siguiente enlace https://niftynet.readthedocs.io/en/dev/config_spec.html.

hay que adaptar una serie de parámetros en la configuración:

- Número de clases. En este caso se fijan en dos: para clasificar la zona lesionada, por un lado, y el resto, por otro.
- Número de iteraciones y cada cuántas guardar un *checkpoint*. Esto se decide y va variando para cada entrenamiento.
- *spatial_window_size*. Se trata del del tamaño configurado para la revisión y/o redimensión de las imágenes, de lo que se hablaba antes.
- *window_sampling*. Aquí se determina que queremos redimensionar, respecto al tamaño de ventana anterior.
- Ejes. Sencillamente los A, R, S de los que se compone una TC.
- Orden de interpolación. Se decide que sea 1 (linear), tras aprender que es un equilibrio entre precisión y procesamiento.
- Función de pérdida. Se mantiene conforme al modelo base (dice), sobre la que también hay buenos resultados según se puede leer en otros estudios. Para el futuro podría ser interesante plantear pequeños ajustes aquí y ver cómo afectan.
- Tamaño del *batch*. Cuántas imágenes se pueden pasar a cada GPU en cada iteración. Originalmente estaba en 6 u 8 y tratamos de mantenerlo así, pero desgraciadamente nos llevó a errores por ser la memoria de la máquina usada para el entrenamiento insuficiente y finalmente tuvo que mantenerse en 2-3 según el entrenamiento y la red (dato que confirmó el mayor uso de recursos de V-Net no densa, que nunca pudo usarse con un batch 3).
- Longitud de cola. Teniendo en cuenta un par de pruebas, el resto del diseño y el bajo número de imágenes disponibles no era relevante su modificación, por ello se mantuvo en el mínimo ($batch * 5$) para priorizar el ahorro de procesamiento.
- Datos específicos del procesado, como el número de GPU's e hilos, delimitado a 3 por la arquitectura de la máquina utilizada.
- Rutas de búsqueda para el acceso a datos de entrada y al modelo, así como otra de guardado de los datos de salida.
- Aplicación. En nuestro caso la de segmentación.

Además de esto, destacar que el archivo de configuración inicial está organizado en diferentes secciones:

- Configuración de datos entrada: donde también se fija un texto como contenido del nombre de los ficheros de cada tipo, de manera que el sistema identifique cuáles son los pares de datos que se corresponden con TC y máscara de un mismo caso. Por tanto habrá 3 subsecciones: para entradas de TC, máscaras y ficheros segmentados, para su evaluación.
- Configuración del sistema. Todo lo referente a la red y variables del entrenamiento, segmentación o evaluación.
- Configuración personalizada de la aplicación, en este caso para la segmentación.

Aparte de esto en la siguiente sección se explica cómo proceder a instalar NiftyNet y ejecutar los procesos.

D.4. Compilación, instalación y ejecución del proyecto

Disponemos de una máquina a la que acceder a través de un servidor, en modo remoto, mediante SSH. En concreto, se accede a través de una línea de comandos con la siguiente estructura: *ssh -X -p 22 usuario@IPservidor*. Así, se indica que se van a ejecutar programas con interfaz gráfica (parámetro '*-X*'), utilizando el puerto 22 ('*-p 22*').

Una vez dentro, hay que se instala NiftyNet y un entorno virtual de Jupyter, si es la primera vez. Este proceso se explica en el fichero **EntornoVirtual&&InstalarNiftynet&&ArrancarJupyter.ipynb**.

Si ya está instalado se ejecuta cada proceso con su comando, tal como se muestra en la memoria:

```
! net_segment train -c /home/bsc0001/niftynet/extensions/dense_vnet_TC_craneal/config.ini
```

Figura D.1: Comando para la ejecución de un entrenamiento.

```
! net_segment inference -c ~/niftynet/extensions/dense_vnet_TC_craneal/config.ini
```

Figura D.2: Comando para la ejecución de una inferencia o segmentación.

```
! net_segment evaluation -c /home/bsc0001/niftynet/extensions/dense_vnet_TC_craneal/config.ini
```

Figura D.3: Comando para la ejecución de una evaluaciónde la segmentación.

D.5. Pruebas del sistema

Esta información también se encuentra resumida en la sección de aspectos relevantes del proyecto de la memoria. Las pruebas que se han llevado a cabo han sido sobre todo de diferentes entrenamientos, mientras que los procesos de segmentación y evaluación nos servían sobretodo para saber que funcionaba y cómo de bien, así como comparar unos entrenamientos con otros.

Entrenamiento

Los entrenamientos más relevantes que se hicieron fueron los siguientes:

- El primero con más de dos imágenes (14) del mismo tamaño entre sí y coincidiendo también con el de la ventana (32 cortes), cuyos resultados ya no resultan en máscaras vacías y se aproximan muy ligeramente, pero más de lo esperando para una muestra de datos tan ínfima. Supuso 7 horas para 1500 iteraciones.
- Uno intermedio con 48 imágenes, de las cuales 14 no tenían lesión (máscara vacía) y las 34 restantes sí. Eran del mismo tamaño, pero esta vez 30 cortes, mientras que la ventana se mantuvo en 32 y por ello se pasaba por un proceso de redimensionado. Con 3000 iteraciones, no mejoraba los resultados anteriores.
- El final, con el máximo número de imágenes que se pudo recopilar y procesar (59), de distintos tamaños (redimensión a 512x512x32) y que se acerca vagamente al objetivo final, algo que tampoco se podía esperar con un entrenamiento tan pequeño para el caso que nos ocupa, pero que sirve experimentalmente para abrir más líneas en las que trabajar en el futuro al atisbar alguno de los problemas que puede estar teniendo, a la par que para mostrar que, aunque poco, algo está aprendiendo, tras 5000 iteraciones en 32h.

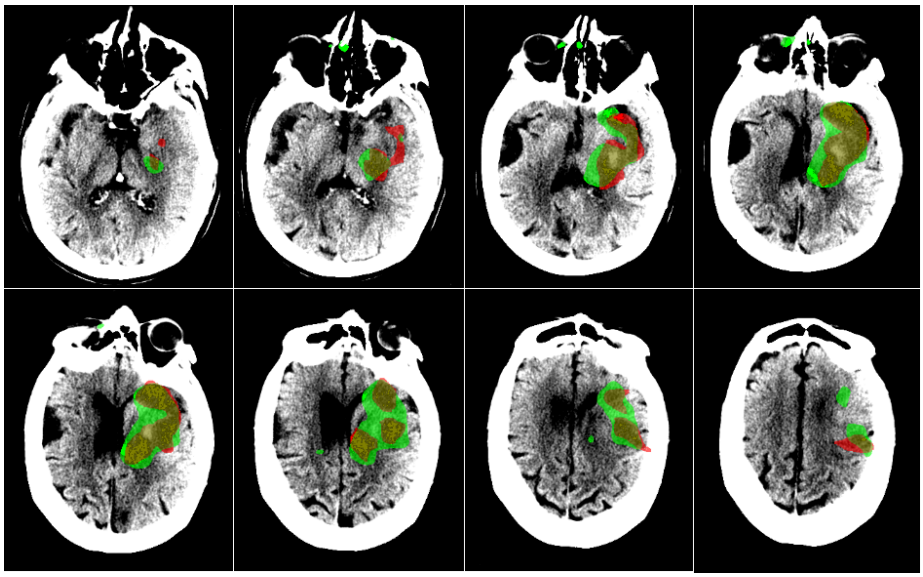

Figura D.4: Resultado del entrenamiento final con una TC conocida. En rojo se representa la lesión y en verde la aproximación del sistema.

En cuanto al entrenamiento final, apenas mejora en segmentación de imágenes conocidas respecto al primero, de hecho hay una imagen en concreto en la que parece ser menos preciso, sin embargo, como se verá en el siguiente apartado sobre la segmentación, sí parece realizar esta tarea con más eficacia (máxima en la figura del siguiente apartado) y sentido, aunque no en todas las imágenes, por desgracia.

Los intentos de entrenar una gran muestra con V-Net (no densa) se vieron frustrados por falta de recursos, agotando el sistema la memoria de las GPUs. Sin embargo, para que quede constancia, esto se llevaría a cabo cambiando la línea del fichero de configuración para ello: Dentro de la sección [NETWORK]

- name = dense_vnet_TC_craneal.CNN_TC_craneal_ictus.MyNet (red propia básica)
- name = dense_vnet
- name = vnet

Segmentación

Una vez se completa el entrenamiento, la segmentación no tiene misterio. Se trata simplemente de organizar el contenido de los directorios para realizar la inferencia sobre los datos que se desee y ejecutar la siguiente celda:

```
! net_segment inference -c ~/niftynet/extensions/dense_vnet_TC_craneal/config.ini
```

Figura D.5: Comando para la ejecución de una inferencia o segmentación.

Entonces, el sistema nos va a devolver un fichero NIfTI comprimido (*.nii.gz*) por cada TC que haya encontrado en la carpeta correspondiente a los datos de entrada. Este es el resultado de la inferencia, tras aplicar los pesos del modelo indicados (*.ckpt*) a la imagen de entrada. Dicho fichero de salida tiene exactamente el mismo aspecto que la máscara realizada manualmente con la ayuda de un experto, es decir, será binario y abierto por sí solo se verá negro con marcas blancas donde estima que puede haber lesión.

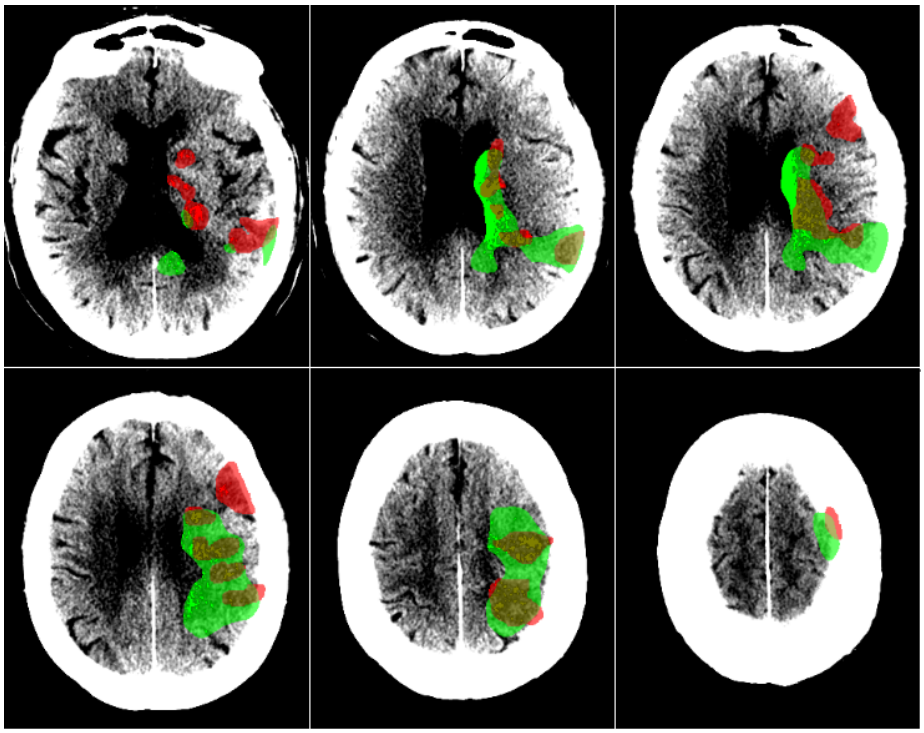

Figura D.6: Segmentación de otra TC craneal con los pesos del modelo de entrenamiento final. La lesión marcada en rojo y la aproximación del sistema en verde.

Como se puede apreciar en la imagen anterior, ambas máscaras se pueden superponer a la imagen de la TC original, de forma que podremos ver dónde se localiza el daño (marcado en rojo) y dónde estima el identificador que está (verde en este caso). Parece que se reitera con esta segmentación la tendencia del sistema a unificar el volumen marcado, probablemente por asociación de ideas y probabilidad, ya que este tipo de lesión es menos habitual. Aclarar que esta figura no recoge una TC craneal completa, sino sólo cortes relevantes, donde hay lesión.

Evaluación

Finalmente, se procede a la evaluación de los resultados de forma más analítica que simplemente mirando la comparativa de las imágenes. Para ello, ejecutamos la instrucción siguiente:

```
! net_segment evaluation -c /home/bsc0001/niftynet/extensions/dense_vnet_TC_craneal/config.ini
```

Figura D.7: Comando para la ejecución de una evaluaciónde la segmentación.

En la sección de datos de entrada del fichero de configuración inicial, deberá haber un apartado para la evaluación, indicando la ruta donde se han ubicado las imágenes a evaluar. Es importante, que si en la sección personalizada no está comentada la línea que hace referencia a la sección para definir la entrada de imagen, el fichero con el CT también esté localizable. Ya que, aunque no es necesario para la evaluación, si se indica en la configuración que se busque dará un error si no se encuentra.

Aparte de todo esto, en las secciones correspondientes al sistema debe haber un apartado para la evaluación, en el que se indiquen qué tipo de datos y evaluaciones queremos obtener, así como en qué unidades se evalúa. Esto último en nuestro caso es *foreground* (primer plano en inglés), ya que sólo hay dos clases, luego basta con evaluar las unidades marcadas respecto al fondo vacío.

A continuación se puede ver el resultado de otra segmentación de imagen desconocida, con una lesión bastante grande, tanto horizontal como verticalmente, para lo que estamos acostumbrados nosotros y el sistema, considerando nuestras muestras. En este caso, la aproximación se queda claramente corta, pero al menos sí acierta con la zona.

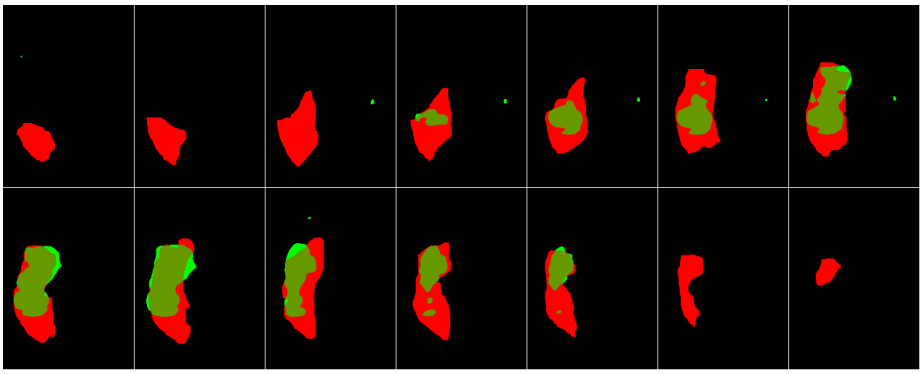

Figura D.8: Evaluación de máscaras tras la segmentación por superposición de la lesión etiquetada por un experto (rojo) y la inferencia a evaluar (verde).

Tal como se trata de representar en la figura D.8, el sistema de evaluación de una segmentación solo toma en cuenta la superposición de las dos máscaras. En este caso, una vez más, la roja es la lesión y la verde la aproximación a

su identificación. Basado en esa superposición crea una matriz de confusión que compara los resultados y operando sobre sus elementos devuelve valores numéricos interesantes para la evaluación, tanto a nivel de cada sujeto segmentado, como la media de todos los de una misma segmentación. El fichero de salida es un *.csv*, que pasado a una hoja de cálculo y un poco editado queda tal como se ve en la siguiente figura:

subject_id	n_pos_ref	n_pos_seg	tp	tn	fn	fp	false_positive_rate	accuracy	sensitivity	positive_predictive_values	negative_predictive_values	jaccard
18_	15996	49201	15224	8338635	772	33977	0,0041	0,9959	0,9517	0,3094	0,9999	0,3046
19_	206275	56706	10479	10233258	195796	46227	0,0045	0,9769	0,0508	0,1848	0,9812	0,0415
65_	122132	45544	42058	7738702	80074	3486	0,0005	0,9894	0,3444	0,9235	0,9898	0,3348
66_	27384	42942	16470	7810464	10914	26472	0,0034	0,9952	0,6014	0,3835	0,9986	0,3058
67_	24708	35016	21158	7825754	3550	13858	0,0018	0,9978	0,8563	0,6042	0,9995	0,5486
68_	219	9928	194	7854367	25	9734	0,0012	0,9988	0,8858	0,0195	1,0000	0,0195

Figura D.9: Resultados de la evaluación por sujeto de la segmentación basada en el entrenamiento final. El sujeto resaltado corresponde con una TC usada en el entrenamiento.

Por completar con algún dato promedio del entrenamiento en general, la media de precisión es del $99,23\,\%$ mientras que en sensibilidad es un $61,51\,\%$. El valor de la función *jaccard* se quedan en $0,26$.

Se confirma con estos datos que un dato de entrada será mejor segmentado que uno desconocido, sin embargo, la diferencia no es tan alta como en el caso del primer entrenamiento. Aparte de la función *jaccard* (intersección de las máscaras partida por su unión), que no tiene traducción como tal, el resto son datos numéricos listados en inglés, el idioma de la plataforma. Para una mayor comprensión los cito y explico brevemente de izquierda a derecha:

- Identificador del sujeto.
- Puntos marcados como lesión por la referencia.
- Puntos marcados como lesión por la segmentación.
- Positivos correctos (lesión bien identificada).
- Negativos correctos (resto bien marcado).
- Falsos negativos (lesión no identificada).
- Falsos positivos (identifica lesión donde no la hay).
- Ratio de falsos positivos.
- Precisión (porcentaje de acierto, sea lesión o no).

- Sensibilidad (porcentaje de acierto, fijándose solo en la zona de dañada).
- Predicción positiva (cuánto es lesión de lo que predice como tal).
- Predicción negativa (cuánto de lo que dice que no es lesión, realmente no lo es).

Apéndice E

Documentación de usuario

En este caso no se ha diseñado una aplicación para el usuario, sino que se trata de un proyecto de aproximación experimental. Por ello, todo el uso se ha explicado en el manual del programador.

Bibliografía

[1] Liang Chen, Paul Bentley, and Daniel Rueckert. Fully automatic acute ischemic lesion segmentation in dwi using convolutional neural networks. *NeuroImage: Clinical*, 15:633–643, 2017.

[2] Eli Gibson, Francesco Giganti, Yipeng Hu, Ester Bonmati, Steve Bandula, Kurinchi Gurusamy, Brian Davidson, Stephen P Pereira, Matthew J Clarkson, and Dean C Barratt. Automatic multi-organ segmentation on abdominal ct with dense v-networks. *IEEE transactions on medical imaging*, 37(8):1822–1834, 2018.

[3] Islem Rekik, Stéphanie Allassonnière, Trevor K Carpenter, and Joanna M Wardlaw. Medical image analysis methods in mr/ct-imaged acute-subacute ischemic stroke lesion: segmentation, prediction and insights into dynamic evolution simulation models. a critical appraisal. *NeuroImage: Clinical*, 1(1):164–178, 2012.

[4] Stefan Winzeck, Arsany Hakim, Richard McKinley, José AADSR Pinto, Victor Alves, Carlos Silva, Maxim Pisov, Egor Krivov, Mikhail Belyaev, Miguel Monteiro, et al. Isles 2016 and 2017-benchmarking ischemic stroke lesion outcome prediction based on multispectral mri. *Frontiers in neurology*, 9, 2018.