# Implementació d'un mòdul descodificador per un sistema OCR

Oriol López Guerra

**Resum**— En el reconeixement de text en escenes, avui en dia retallar la paraula amb el correcte significat segueix sent un fet complicat. En aquest treball veurem una de les xarxes que ha obtingut bons resultats. Utilitzarem la xarxa BI-STET que utilitza CNN i transformers encoder i decoder. Veurem com funciona i intentarem introduir-li una CNN diferent anomenada PHOCNet, aquesta a diferencia de la BI-STET està pensada per treballar amb imatges amb lletres escrites a mà. En aquest treball compararem les dos CNN esmentades i veurem les seves possibilitats, avantatges i diferencies.

**Paraules clau**— Pytorch, Python, xarxes neuronals, Intel·ligència artificial, tensorflow, reconeixement de text, BI-STET, PHOCNet, Resnet, transformer.

**Abstract**— Nowadays, In Scene Text Recognition (STR), crooping the words with the correct meaning is still complicated. In this work we will see one of the network that has obtained good results, we will use the BI-STET network that uses CNN and encoder and decoder transformers. We'll see how it Works and try to introduce a new diferent CNN called PHOCNet, unlike BI-STET is designed to work with images with handwritten letters. In this paper we will compare the two CNN mentioned and see their possibilities, advantages and differences.

**Index Terms**— Pytorch, Python, Neuronal Networks, Machine learning, tensorflow, Scene Text Recognition, BI-STET, PHOCNet, Resnet, Transformer.

## 1 Introducció

### 1.1 Concepte general

La visió per computadors des de fa ja anys està guanyant molt de protagonisme en el món de l'enginyeria. Gràcies a aquesta i altre conjunt de tècniques podem adquirir, processar, analitzar i comprendre per mitjà d'imatges el món real, transformant-lo a una representació numèrica que pot ser tractada amb els ordinadors. Al mateix temps el deep learning també ha anat progressant. Han sorgit xarxes neuronals com les CNN, les quals permeten extreure característiques de les imatges, poden així classificar-les. Junt amb aquest tipus de xarxes cap al 2018 van sorgir les xarxes transformer utilitzades en gran part per la traducció de textos en idiomes, l'avantatge d'aquest tipus de xarxes és el fet que són capaces de processar una seqüència molt més gran de paraules de cop.

En aquest treball proposat per l'Ernest Valveny Llobet aprofitarem aquestes noves tecnologies, per estudiar i treballar amb aquest tipus de xarxes. Convertirem imatges de paraules de la vida real, en aquesta representació numèrica i d'aquí extreure el seu significant, també mirarem com es podrien millorar els resultats obtinguts.

Durant aquest article parlarem sobre els nostres objectius proposats i a on volem arribar, explicarem com ha estat format el treball, així com components, datasets i més detalls importants del treball. Mostrarem la planificació que hem seguit i com ens hem gestionat el temps, per acabar explicarem el desenvolupament del treball i els resultats extrets.

### 1.2 Objectiu general del projecte

L'objectiu general del projecte consisteix a entrenar una xarxa neuronal per poder extreure el text que hi hagi en fotografies. Per poder realitzar-lo es necessari marcar-se un seguit d'objectius, aquets van predeterminats amb les tecnologies esmentades anteriorment i els objectius de treball que exposarem més endavant.

La idea principal es analitzar una xarxa basada en transformers, són xarxes les quals funcionant molt bé pels nostres objectius, gràcies a que disposen de codificadors i descodificadors, els quals faciliten la interpretació de les paraules. S'ha escollit la xarxa Bi-STET que és simple i eficient, l'objectiu serà analitzar aquesta xarxa, provar-la, executar-la i fer-li certes modificacions a la part del codificador per analitzar si podem millorar el seu rendiment.

La Bi-STET té dues parts: una CNN per extreure les seves característiques i després la transformer per obtenir la transcripció de les dades obtingudes anteriorment a la CNN en la paraula resultant.
Per tant el primer objectiu ha sigut informar-se i relacionar-se amb les xarxes CNN amb la llibreria Pytorch.
Un cop assimilat aquest objectiu podríem passar a la part de les xarxes transformer les quals seran capaces de traduir les imatges amb les seves característiques a paraules, el segon objectiu seria familiaritzar-se amb les xarxes transformar i executar-ne una.

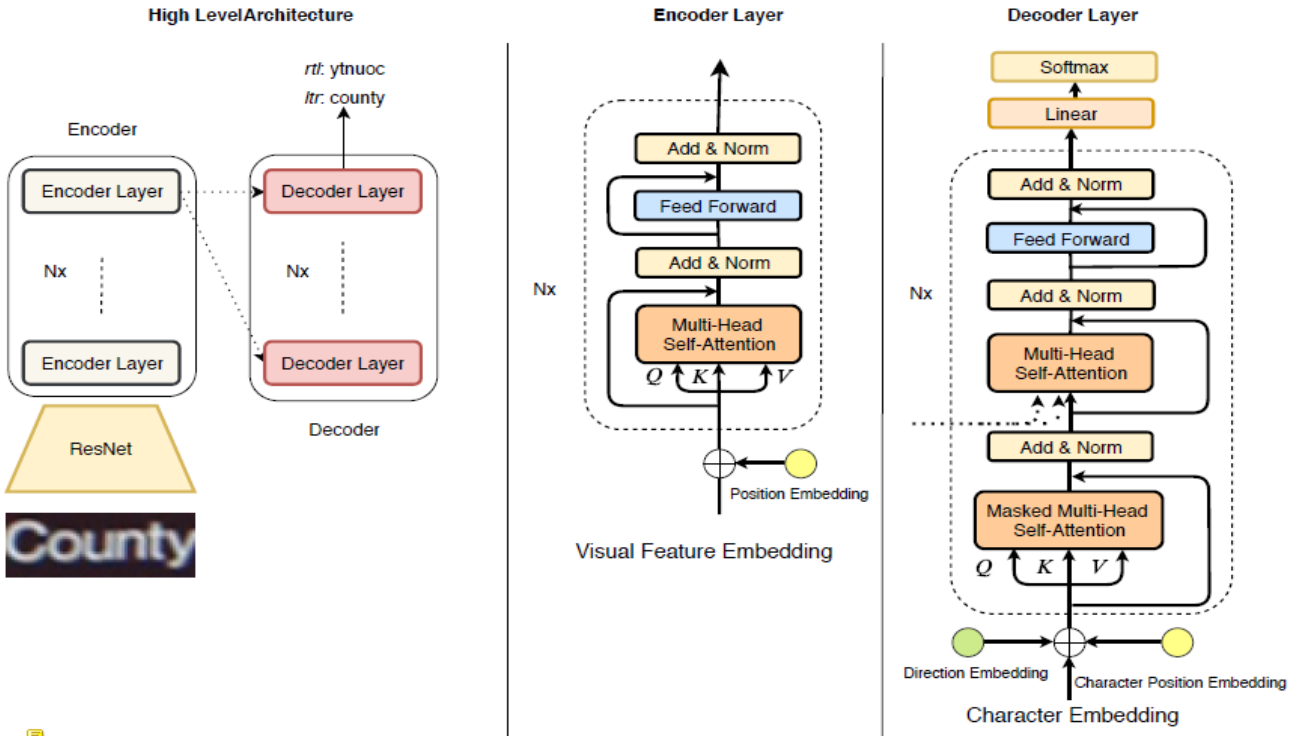
La part final del nostre projecte consisteix en canviar la xarxa CNN que utilitza la nostra xarxa transformer per una CNN que és capaç de guardar les dades en un format de codificació diferent seguint el projecte PHOCNet. La PHOCNet és una xarxa preparada per treballar directament amb paraules, capaç d'acceptar imatges d'entrada amb mides arbitraries, sense necessitat de redimensionar ni retallar les imatges, transformant aquestes en una representació codificada de la paraula. Amb aquest format podem intentar implementar una detecció de paraules per valors la qual podria ser més ràpida. Per tant tindríem un últim objectiu , que seria el anàlisis i comparació del resultats de la nostra nova xarxa respecte el prototip transformer Bi-STET.
Possibles objectius ja més petits en el cas de tenir temps o altres factors, seria intentar millorar resultats obtinguts.

## 2 Estat del art

Per realitzar aquest projecte ens hem basat principalment en dos articles.

L'article Bidirectional Scene Text Recognition with a Single Decoder [3], o anomenat també Bi-STET, com podem llegir en el seu article, és una xarxa que ha donat molts bons resultats en el reconeixement de text i té unes petites modificacions respecte a les altres xarxes transformer. Està formada per diverses xarxes principalment una Resnet [10], n codificadors i n descodificadors, com a resultat final una sortida. La primera seria una ResNet utilitzada per obtenir les característiques de les imatges per mitja de patrons. D'aquí trauríem un embedding amb les seves característiques guardades en aquest vector i li passaríem un embedding de posició, aquest seria passat com a input als n codificadors. Seguidament es passarien els embeddings resultats pels n descodificadors i s'aplicaria el de posició i un embedding de direcció, és aquí on recau el principal canvi ja que aquest element fa que es realitzi d'esquerra-dreta o dreta-esquerra el text. Tot això és el input dels n descodificadors i obtenim les paraules resultants per mitja de una capa linear i un softmax. Podem apreciar l'arquitectura a la il·lustració 1.

Per la PHOCNet [2], tenim que és una xarxa que ens permet diferents mides de dades d'entrada, d'aquesta manera treballa sense redimensionar la imatge ni perdre característiques i detalls importants del text. La codificació de la paraula és fa en vectors de característiques de valor 1 i 0. Es fan divisions de les paraules en aquests vectors esmentats, dintre d'aquestes divisions tenim les lletres corresponents de la paraules representades amb les seves característiques. Amb aquest tipus de codificació s'obté informació especifica de tota la paraula, lletra a lletra. Ja que codifica els caràcters que apareixen a la paraula i en quines posicions de la paraula estan aquests.
És per aquest motiu que hem escollit aquesta xarxa, per que ens permetrà obtenir característiques més ràpidament.
L'esquema de la PHOCNet el podem veure il·lustració 2.

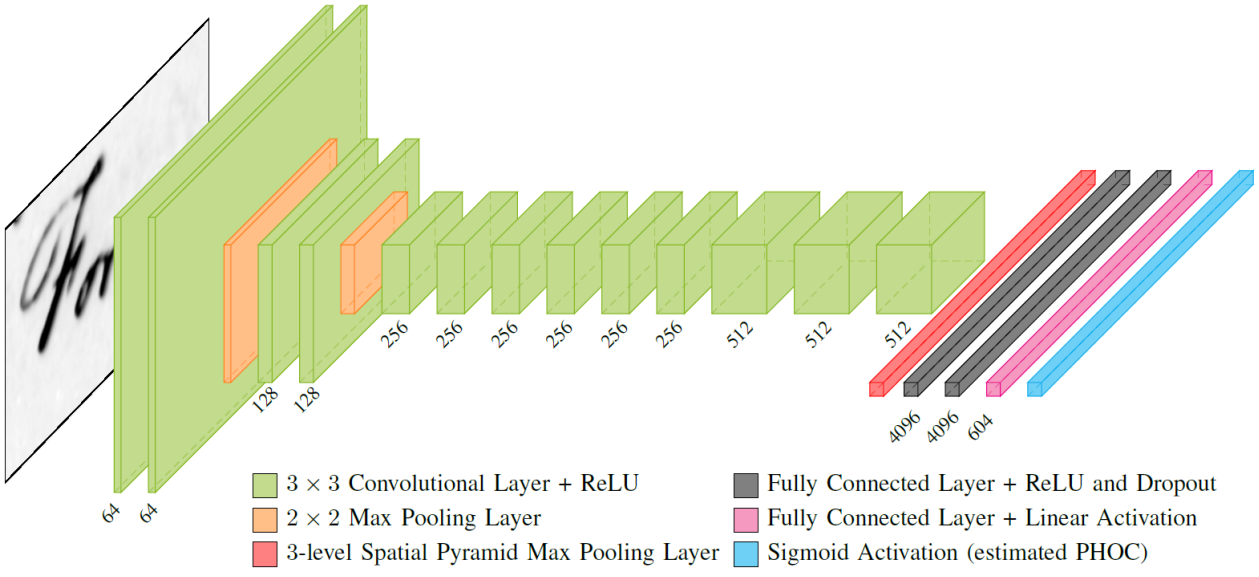



*Il·lustració 1: Model BI-STET*


- *E-mail de contacte: oriol.lopezgu@gmail.com*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Ernest Valveny Llobet (Ciències de la computació)*
- *Curs 2020/21*


Per aquestes dos xarxes hem obtingut el seu codi públic del repositori github [24] [25].

Les nostres xarxes treballen amb la llibreria pytorch [1], a més el tutorial que vam seguir per repassar les CNN està disponible a la




*Il·lustració 2: Model PHOCNet*

seva web, així com molta informació sobre la llibreria i altra informació rellevant envers el nostre projecte.

La PHOCNet per ferla servir amb pytorch hem utilitzat una versió feta per un alumne de doctorat de la Universitat Autonoma de Barcelona, fer menció especial aquest alumne [13].

Per acabar fer una petita menció a que tot el treball està realitzat amb el llenguatge de programació python [9] un dels més utilitzats avui en dia per realitzar aquest tipus de feines.

## 4 EINES I TECNOLOGIES UTILITZADES

El projecte es planteja amb el llenguatge de programació python, com a referencia a ser un dels mes utilitzats per treballs de deep learning.

Per treballar amb python s'ha instal·lat anaconda amb l'eina Spyder que facilitarà el treball amb aquest llenguatge, així com algunes instal·lacions de llibreries. També hem creat un entorn virtual dintre anaconda, així podem instal·lar les llibreries necessàries pel moment i després eliminar-ho tot sense preocupacions d'espai.
També es va considera utilitzar l'eina google colab, aquesta eina esta preparada per treballar en deep learning, proporciona als usuaris una GPU o CPU per executar els codis, té un funcionament semblant a l'eina Jupiter. Finalment hem fet servir nomes python amb anaconda ja que ens era més còmoda.

Com que hem treballat també amb una gràfica proporcionada per la Universitat Autònoma de Barcelona. Ja que utilitzem windows nosaltres hem utilitzat una connexió via ssh per connectar-nos i una SFTP per intercanviar arxius, mobaxTerm o putty i winSCP o fileZilla corresponentment.

A l'hora de treballar amb la gràfica des del nostre ordinador hem hagut d'instal·lar els divers més recents de la nostra gràfica i cuda, així podíem executar el codi amb la nostra gràfica.

### 4.1 Llibreries utilitzades

Les llibreries instal·lades en el entorn virtual creat en anaconda han sigut les següents:

- La llibreria Tensorflow [4]
- La llibreria torch [5] i torchvision [6]
- La llibreria tensorboard de pytorch [11]
- La llibreria seaborn [7]
- La llibreria numpy [12]

Amb aquestes hem solucionat tot el problema de dependències.

### 4.2 Datasets

Per poder entrenar la nostra xarxa es necessari utilitzar datasets amb imatges. En el article de la BI-STET utilitzant dos datasets d'imatge per entrenar la xarxa, un d'ells conte un número excessivament elevat d'imatges, així que nosaltres utilitzarem només el Synth90K [14]. Aquest conte un total de 90.000 paraules, totes les imatges són petits cartells amb les paraules escrites, amb cal·ligrafia més semblant a impremta o màquina, no manual.
Per un altre costat el dataset que utilitza la PHOCNet és un data set més semblant a lletra escrita a mà, aquest és el IAM Handwriting Database [15].

Per un altre costat, per assegurar-nos de que la nostra xarxa ha tingut un entrenament correcte utilitzarem datasets de validació.

- ICDAR03 [16}: Conte 218 imatges i 1,156 paraules creades per detecció i reconeixement.
- ICADR13 [17]: Conte bastantes imatges del ICDAR03, però aquestes les retallaren un 15%.
- ICDAR15 [18]: Consta de 2,077 imatges, estan retallades de vídeos recopilats amb google glass. Per tant contenen soroll, aspectes borrosos pel moviment i il·luminacions.
- SVT [19]: Les imatges d'aquest data set han sigut extretes de Google Street View, per aquest motiu les

imatges contenen poca resolució i soroll.
- SVTP [20]: Són 645 imatges retallades del data set SVT.
- IIIT-5K Word [21]: Conte 3,000 imatges, retallades de escenes i imatges digitals.
- CUTE80 [22]: Les imatges d'aquests datasets són corbades o orientades.

## 5 Requisits

Per realitzar aquest treball hem utilitzat certs requisits de software i altres de Hardware.

### 5.1 Requisits del software

- Windows 10 o més recent, macOS 10.10 o més recent, Linux (64-bit o 32-bit x86)
- Hardware que suporti Anaconda amb Spyder i llibreries instal·lades.

### 5.2 Requisits del hardware

Els requisits mínims de la meva màquina, tenint en conta que només hem compilat amb la targeta gràfica i que ha pogut extreure resultats són:

- Qualsevol CPU de 5 o 7 generació corresponent a intel amb AMD el seu corresponent equivalent, 6 cores poden reduir a 4 cores perfectament.
- Memòria ram 16 GB es podria amb 8 perfectament.
- Emmagatzemant hem necessitat uns 60 GB aproximadament pels datasets, llibreries i programes.
- Targeta gràfica GeForce GTX 1060 3GB GDDR5

En el cas de ja buscar uns resultats més ràpids i millors passaríem a utilitzar:

- Targeta gràfica GeForce GTX Titan X, similar o superior, important que disposi de aproximadament uns 5GB o superior, per poder emmagatzemar les dades.
- La resta de components funcionaria amb els esmentats anteriorment, si es poden millorar hi haurien petits beneficis.

## 6 Planificació del projecte

S'han dividit diferents tasques del projecte en diferents parts, aproximadament mensualment, concordant amb les entregues. Com que hi ha hagut parts que han sorgit riscos trobats hem hagut de fer petites modificacions per adaptar-nos.
Per repartir el temps de treball i la feina a realitzar s'ha creat un diagrama de Gantt que podem veure a la il·lustració 4, ens permet portar una bona gestió del temps treballat i necessari per realitzar cada objectiu.

- **Familiaritzar-se amb l'entorn i lectura dels articles**: La fase inicial del projecte es per familiaritzar-se amb els diferents tipus de xarxes, practicant amb el tutorial de pytorch [8], instal·lant i investigant les eines esmentades anteriorment. Fer les corresponents instal·lacions del entorn. També llegint, resumin i destacant el contingut útil dels articles, així com altres fonts d'informació.

- **Execució de la xarxa transformer**: Anàlisis del codi de Bi-STET, entendre el seu funcionament. Realitzar petites modificacions per adaptar-lo a les nostres necessitats, finalment posar el codi en funcionament. Si sobra temps es pot començar a implementar la CNN.

- **Creació de la CNN**: Implementació i finalització de la CNN seguint el format PHOCNet, ajuntar-ho amb la xarxa transformer i començar a extreure resultats amb els datasets desitjats. Com veurem han sorgit certs problemes que fan que aquesta part en el diagrama sigui més extensa.

- **Creació i Anàlisis de resultats**: Fase final del projecte on generem els resultats obtinguts i es compararan amb el els resultats de Bi-STET obtinguts originals i amb la nostra màquina. En un cas de poder optimitzar alguna part del projecte s'aprofitaria si sobres temps o fos possible per implementar i veure les millores.

## 7 Riscos del projecte

En la taula 1 es poden veure els riscos i contingències del projecte.

### 7.1 Riscos sorgits

Durant el nostre projecte ens han sorgit alguns dels riscos esmentats a la taula 1. Vam tenir dificultats en introduir la PHOCNet a la xarxa transformer. Al intentar utilitza la versió del repositori proporcionat pel article [25] aquest està realitzat amb un sub-modul anomenat caffe, el qual funciona amb C. Ens van sorgir molts problemes de dependències i vam procedir a utilitzar el pla de contingència , vam passar a extreure resultats amb la Resnet només. Un cop vam veure que disposàvem de temps suficient ja vam procedir a incorporar la PHOCNet esmentada a la secció 2, que funcionava amb pytorch.

El següent risc amb el que ens vam topar va ser la falta de capacitat HW. La PHOCNet requeria de més memòria ram de la que el nostre ordinador personal disposava, solució utilitzar una màquina de l'Autònoma que ens va proporcionar el nostre tutor.

## 8 Metodologia

La metodologia empleada per dur a terme aquest projecte ha consistit en la metodologia kanban, és una metodologia senzilla d'utilitzar, fàcil d'actualitzar i molt visual. L'eina Trello ens permet fer un bon ús d'aquesta i serà la utilitzada, junt amb alguns pòsits o apunts físics per les tasques més importants, tenir-ho encara visible d'una manera més directa.

Ens permet també adaptar-nos a las tutories i la planificació, a més és incremental, a mesura que s'avança i es veu la situació del projecte es pot anar modificant. Per totes aquestes raons ha estat la utilitzada i ens ha permès una supervisió clara de les feines realitzades.

| Riscs | Impacte | Conseqüència | Probabilitat | Pla de contingència |
|---|---|---|---|---|
| **Dificultats en posar en marxa la xarxa transformer.** | Crític | No es podria seguir amb cap objectiu del projecte. | Probable | Buscar una nova xarxa que sigues igual de competent. |
| **Dificultats en la incorporació de la PHOCNet.** | Fort | Implicaria una part dels objectius fallits. | Probable | Podriem seguir treballant només amb la Resnet i extreure i analitzar resultats. |
| **Temps d'execució massa elevat** | Lleu | No hauria de tenir un impacte gran, ja que no busquen un número d'iteracions tant elevat. | Poc probable | Com que tenim una bona organització no ens hauria de sorgir aquest problema. |
| **Falta de capacitat Hardware** | Moderat | No podríem executar el codi de manera optima. | Probable | Disposem d'una màquina a l'autònoma que ens permetria solucionar aquest problema. |
| **Ser massa ambiciós i mala gestió del temps** | Fort | Podria donar-se el cas que no incorporéssim la PHOCNet i no obtinguéssim resultats. | Poc probable | Seguir la planificació i pensar en els objectius plantejats i els plans de contingència esmentats per superar aquests contratemps. |
| **L'ordinador deixi de funcionar, pèrdua de contingut.** | Crític | Enrederiria el desenvolupament del projecte i no podríem aconseguir tots els objectius. | Baixa | Crear copies de seguretat, tenir el codi penjat a la maquina de l'autònoma. |

*Taula 1: Riscos i contingències del projecte*

## 9 Desenvolupament i Resultats

A continuació s'explicarà tot el desenvolupament del projecte, implementacions realitzades, aprenentatge i procediments realitzats.

Com que durant la realització del nostre projecte era quan es generaven els resultats, aprofitarem per anar-los comentant a mesura que expliquem el desenvolupament del nostre projecte, per no fer-ho tant confús.

Hem utilitzat tots els datasets de validació disponibles, el dataset Synth90K com a training i el IAM Handwriting amb una partició com a training i dos com a validation, com es proporciona a la web d'aquest.

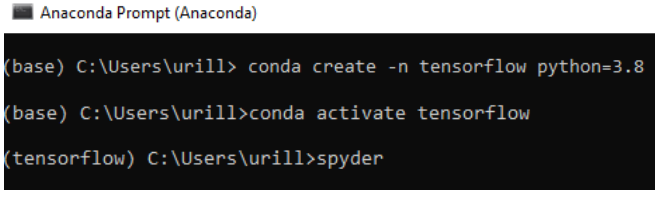
### 9.1 Primer contacte amb pytorch

El nostre primer pas va ser seguir el tutorial proporcionat per pytorch [8] sobre xarxes neuronals i CNN, per tal de repassar els conceptes i veure com funciona pytorch. A més ens vam llegir els articles ja esmentats [2] i [3] per entendrà com funcionaran les xarxes que utilitzarem.

El tutorial el vam realitzar amb google colab, vam veure que funcionava bé, però que per gestionar diferents fitxers i dependències era una mica complicat. Per aquest motiu vam decidir passar a utilitzar un altre entorn per realitzar el projecte.

### 9.2 Instal·lació i configuració de l'entorn de treball.

Un cop abandonat el google colab vam procedir a instal·lar anaconda, ja que disposa de facilitats per instal·lar les llibreries per mitja de l'aplicació, un IDLE per python i ens proporciona poder crear un entorn virtual.




Des de conda promt vam poder instal·lar les dependències esmentades anteriorment. Un cop realitzat això ja podíem començar a treballar en la segona fase del projecte amb l'entorn Spyder.

### 9.3 Descarrega i creació dels fitxers Datasets

Per poder posar en marxa el codi abans vam descarregat tots els datasets comentats en la secció 4.2, el codi de la Bi-STET [24] conte una carpeta data_utils, allà estan els fitxers que generant els path i lexicons corresponents per treballar amb les imatges. El que hem fet es fixar-nos en que els datasets descarregats, es relacionessin amb els establerts en aquell fitxer, un cop assegurat això al treballar amb Windows hem hagut de modificar tots

els path del codi amb os.path.join ja que el codi no treballava amb els path així. Un cop fet tots els canvis hem executat els arxius corresponents per generar el codi.

## 9.4 Execució BI-STET

Un cop ja tenim els datasets amb les imatges, els seus path i lexicons generats, ja vam passar a mirar tot el codi de la Bi-STET, el fitxer principal que hem modificat per entrenar és el config.py, en aquest igual que a la resta del codi, tots els path no estan filtrats per Linux o Windows, per tant hem hagut de retocar tots els path. En aquest fitxer es on canviem totes les configuracions abans d'entrenar o avaluar la xarxa, també on especifiquem el model a carregar si és que volem.
Abans d'executar vam anar mirant tot el codi per trobar coses com on es carregant les imatges, el trainer, la creació de la xarxa, etc.. I a tots els llocs importants vam anar posant punts de parada per veure l'execució pas a pas per quan toques modificar codi o corregir errors.
Després de corregir errors de path i assegurar-nos que arribava fins la part abans d'entrenar, vam decidir executar la primera execució amb les dades originals del article [3]. Com era d'esperar-se la nostra targeta gràfica no podia treballar amb un batch size tant gran. Llavora'ns vam decidir baixar-ho a 1 i modificar el número d'iteracions a 10000 només per provar que funciones. Així com modificar el lr a 0.1, i vam obtenir els següents resultats de right to left, left to right i bidireccional, mostrats a la Taula 2. Un cop vist aquests resultats ja sabem que podem executar el codi, llavora'ns vam passar a realitzar un entrenament ja de 100,000 iteracions amb un batch size de 8 que és el màxim que pot executar la nostra targeta gràfica.

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 27.3 | 28.8 | 39.0 | 30.1 | 15.8 | 19.5 | 13.2 |
| *r-t-l* | 26.3 | 27.2 | 39.7 | 31.4 | 15.4 | 17.8 | 14.2 |
| *bid* | 29.3 | 30.8 | 41.2 | 33.2 | 16.4 | 20.9 | 14.2 |

*Taula 2: Primers resultats 10k iteracions*

### 9.4.1 Resultats BI-STET Original

Els resultats obtinguts del article estan realitzats amb un total de 500,000 iteracions, un batch size de 64, learning rate de 1 que es redueix a les iteracions ja comentades anteriorment, una inicialització de pesos de Xavier-normal i ADADELTA com a optimitzador [28]. Els resultats originals i sobre els que farem comparativa són els següents, tenint en conta que l-t-r és left-to-rifght, r-t-l és right-to-left i bid fa referència a bidireccional:

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 94.2 | 88.3 | 93.8 | 85.1 | 75.2 | 78.8 | 81.8 |
| *r-t-l* | 94.1 | 87.9 | 94.8 | 85.7 | 73.3 | 79.5 | 83.6 |
| *bid* | 94.7 | 89.0 | 95.2 | 85.8 | 75.8 | 80.6 | 82.5 |

*Taula 3: BI-STET Original 500k iteracions*

Com podem veure els resultats originals són bastants grans, a més realitzen moltes iteracions factor que no te per que ser tant important. Però si que veiem que respecte a la nostra primera execució Taula2 són molt superiors.

### 9.4.2 Resultats màquina personal

Els resultats generats amb la nostra màquina i tots els altres apartir d'aquest punt van ser amb un únic dataset de training com bé s'ha comentat anteriorment. Hem utilitzat paràmetres bastants similars als anterior. Hem modificat únicament el learning rate baixant directament a 0.1, el número d'iteracions a 100,000 i batch size a 8. Aquests canvis com ja hem explicat són per intentar obtenir resultats acceptables amb un temps més reduït, els resultats obtinguts són els següents:

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 68.2 | 65.7 | 79.1 | 67.0 | 44.9 | 49.0 | 38.0 |
| *r-t-l* | 66.3 | 65.2 | 77.6 | 65.8 | 45.1 | 47.4 | 39.0 |
| *bid* | 69.0 | 67.2 | 79.4 | 67.0 | 47.1 | 51.2 | 39.7 |

*Taula 4: Màquina personal 100k iteracions*

Podem veure que ja obtenim resultats bastants més elevats respecte a la taula 2 i que ens comencem a apropar a resultats més similars a la original. Però veiem que els datasets amb més soroll, o imatges retallades o modificades respecte les originals costa més obtenir resultats, segurament és necessari més iteracions de training, però el fet de reduir el lr a 0,1 ens ha proporcionat resultats acceptables ràpidament. Ara intentarem millorar aquests amb la màquina de l'Autònoma.

### 9.4.3 Resultats màquina Autònoma

Com que la màquina de l'autònoma té una memòria més gran podem provar la configuració anterior però pujant el bach size a 64 com la original.
Hem pogut veure una gran millora al incrementar el número d'imatges per iteració d'entrenament:

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 61.4 | 62.0 | 74.4 | 62.4 | 43.6 | 46.7 | 35.8 |
| *r-t-l* | 60.2 | 61.5 | 74.2 | 60.5 | 42.9 | 46.4 | 37.8 |
| *bid* | 62.5 | 64.1 | 75.6 | 62.8 | 44.1 | 48.5 | 39.6 |

*Taula 5: Màquina Autònoma 20K iteracions*

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 76.1 | 73.9 | 86.5 | 72.4 | 54.3 | 59.8 | 56.6 |
| *r-t-l* | 74.8 | 73.0 | 86.9 | 72.5 | 52.8 | 57.4 | 55.6 |
| *bid* | 76.9 | 75.4 | 87.8 | 73.5 | 55.3 | 61.7 | 57.6 |

*Taula 6: Màquina Autònoma 100k iteracions*

Al [A2] podem veure la diferencia que obtenim respecte el original utilitzant el dataset amd reducció en 100k iteracions.

Com podem veure i comentarem a les conclusions, els resultats obtinguts amb el batch size augmentat és molt notori, fins el punt que sembla que la gràfica convergeix molt. Per altre banda decrementar el learning rate manualment a certes iteracions no ens aporta grans canvis amb aquest número total d'iteracions.

## 9.5 Incorporació de la PHOCNet

Amb els resultats que vam obtenir en el apartat anterior, vam procedir a modificar la Resnet per la PHOCNet.

Un cop ja tenim la Bi-STET funcional van començar a mirar com incorporar la PHOCNet, el codi d'aquesta es troba en el github

[25]. Aquesta xarxa està composada per un submòdul anomenat caffe, per tant per fer-la funcionar era necessària la instal·lació d'aquest.
La instal·lació es realitzava mitjançant C i Make, per poder dur a terme en Windows aquesta instal·lació vam haver d'instal·lar diferents dependències. La primera era git per extreure el contingut del directori oficial. Com que és un submòdul optimitzat fa servir cuda que ja el teníem instal·lat, cuDNN i nvidia tooltik, finalment per poder executar els fitxers make vam haver d'instal·lar un programa anomenat Cmake [26] i ninja [27]. Al estar realitzat amb C necessitàvem un compilador per aquest llenguatge, en el nostra cas va ser el Visual Studio. Un cop instal·lats tots els programes demanats pel submòdul caffe, vam procedir a la instal·lació d'aquest. Però amb el que ens vam trobar va ser una constants d'errors de dependències i problemes d'execució, per tant no podíem utilitzar la PHOCNet amb aquest submòdul.

Parlant amb el nostre tutor vam decidir canviar, i utilitzar la versió de la PHOCNet esmentada anteriorment creada per un alumne de doctorat, la qual esta realitzada amb pytorch [13]. Amb aquesta versió hauria de ser més fàcil integrar-la ja que no contenia dependències.

Com bé hem dit abans la PHOCNet està pensada per entrenar imatges en grisos, per tant al introduir-la hem de fer petites modificacions en el input. Això es degut a que li entren ara 3 dimensions i no 1. Hem d'ajustar el output a 512 que és amb el que treballa el decoder, i la sortida al forward la podem deixar igual o la podríem redimensionar com es fa a la Resnet, nosaltres la deixarem igual. També afegim al fitxer config.py opcions per triar Resnet o PHOCNet a l'hora de crear o carregar la xarxa.
Un cop introduïda vam procedir a executar i mira els resultats obtinguts. El Mean loss ens pot donar una idea de si el nostre entrenament està funcionant bé, la Resnet apartir de les 3000 iteracions ja ens reduïa a 1 casi 0,9, per un altre costat la PHOCNet no ens baixava en cap moment del 5 això ens feia ja pensar que no obtindríem els resultats esperats. Efectivament els resultats obtinguts van ser de 0 a la validació.

Com que la PHOCNet sabem que treballa amb datasets diferents vam procedir a utilitzar aquests.

### 9.5.1 Provar data set IAM

Vam descarregat el dataset IAM Handwriting [15], agafarem aquest dataset d'imatges i provarem a veure si amb aquest podem obtenir millors resultats que al article [2].
Primer de tot vam descarregar el dataset amb les paraules, imatges i el ASCI per crear el lexicon. Seguidament vam crear un petit script que ens genera els fitxers de training i datasets corresponents "procés_iam.py. Tres particions de 6161 linies de text per training i dos de 900 i 940 com a validacions.
Com amb la resta de datasets hem de gestionar també la carrega d'aquest dataset, afegint els path necessaris per agafar la part de training i la part de validation del dataset.

Un cop fet aquest pas ja ho vam passar a la màquina virtual per executar allà i veure els resultats obtinguts.

### 9.5.2 Dataset IAM resultats.

Un cop vam tenir el dataset IAM vam procedir a provar dos execucions diferents. Agafant aquest dataset només com a training i les dos configuracions que ens ha semblat més optima durant totes les execucions anteriors.
Una execució a estat feta amb la Resnet i hem obtingut els següents resultats:

| | *Partició 1* | *Partició 2* |
|---|---|---|
| *l-t-r* | 72.1 | 70.3 |
| *r-t-l* | 71.2 | 69.7 |
| *bid* | 73.6 | 72.0 |

*Taula 7: RESNET IAM 100K iteracions*

L'altre amb la PHOCNet amb resultats:

| | *Partició 1* | *Partició 2* |
|---|---|---|
| *l-t-r* | 57.4 | 56.4 |
| *r-t-l* | 57.6 | 56.3 |
| *bid* | 60.0 | 59.0 |

*Taula 8: PHOCNet IAM 100K iteracions*

### 9.6 Optimitzacions provades

Vam voler provar a millorar els resultats obtinguts amb la Resnet. Per pensar en com millorar això ens em basat en les dades inicials que utilitzant en el article, ells feien servir un Batch size de 64. Com que la nostra gràfica no podia amb aquesta quantitat de dades, vam passar a realitzar els entrenaments amb la màquina proporcionada ja explicada anteriorment. També com que ells utilitzaven un learning rate de 1 i reduït en 0.1 a les iteracions 150,000 , 300,000 i 400,000. Clarament nosaltres si volem intentar obtenir resultats més ràpids no podem executar-ho així, per tant hem fet proves amb learning rate directament de 0.1 i 0.01. També aquests mateixos amb reducció del learning rate però a iteracions 25,000 , 50,000 i 75,000.
Una possibilitat de millora era canviar el mètode d'inicialització dels pesos, ells utilitzant Xavier inicialització. Es podria provar amb diferents inicialitzacions.
Hi ha altres millores com serien agafar un dataset més gran d'imatges o ajuntar els dos datasets de training com explicant al article, però com estem buscant resultats només amb el dataset de training comentat a la secció 4.2 no provarem aquesta millora.
Com que hem vist que tots aquests canvis no han tingut un gran impacte al treballar amb iteracions baixes respecte les originals. Tampoc vam decidir dedicar-li molt més temps a això i centrar-nos més en la implementació de la PHOCNet.

## 10 AMPLIACIONS

El treball es podria ampliar a futur amb els següents aspectes:

- Fer servir més datasets de training i de validation, així com provar entrenant diferents convinacions d'aquests.
- Modificar la PHOCNet per que s'adapti a treballar amb RGB a la BI-STET o més ideal, adaptar la BI-

STET per que accepti la sortida original de la PHOCNet.
- Provar més paràmetres d'optimització diferents com els que s'han explicat anteriorment.

## 11 Conclusions

Al final, tot i les petites complicacions que han succeït, s'han pogut assolir tots els objectius proposats, en part gràcies al pla de contingència.

Com a resultats obtinguts, com hem pogut anar veient la BI-STET ja està molt ben optimitzada i ha sigut difícil millorar els resultats. Pel que hem vist amb un batch size alt convergia molt ràpid l'aprenentatge així com la gràfica de resultats. Per tant podríem dir que de les millores que hem fet amb la nostra màquina personal, la seva millor optimització és augmentar el batch size o segurament provar amb alguna altre xarxa diferents.
Per la màquina de l'autònoma amb el batch size de 64, com bé acabem de dir les millores són poc notòries i hauríem de treballar amb iteracions molt més altes per poder veure millores o resultats diferents..

Respecte a la PHOCNet hem pogut comprovar que el fet d'estar dissenyada per uns datasets escrits a mà limita molt el seu us a aquests. A més treballar amb imatges en 3 dimensions quan esta pensada per grisos, també empitjora els resultats obtinguts. No veiem gran diferencia entre la incorporació del transformer respecte els resultats del article original. Per altre banda la BI-STET veiem que si l'entrenem amb aquest dataset es capaç de proporcionar resultats propers als del article original de manera ràpida. De totes les possibilitats provades extraiem que amb unes 15k o 20k iteracions és on la gràfica dels resultats ja comença a convergir, per tant el número necessari d'iteracions incrementa molt per millorar els resultats.

Com a conclusió final on esperàvem que la PHOCNet fos millor per com ha estat dissenyada i plantejada, tot i no tenir les millors condicions per la PHOCNet hem vist que la RESNET pot ser igual d'eficient gràcies a la BI-STET.
El fet de modificar tant les imatges, quan la PHOCNet està pensada just per això pot ser un dels factors que fan que els resultats obtinguts no fossin els esperats.

De totes maneres els resultats obtinguts són bastant bons com podem veure a la il·lustració. On els primers són el original i els següents amb dataset reduït i 100k iteracions. I per la PHOCNet també són bastant elevats.

## 12 Agraïments

M'agradaria agrair als meus companys i família pel suport que m'han donat durant tot el treball, així com al meu tutor de projecte, per tota l'ajuda i motivació aportada, m'ha fet el projecte més amè i productiu.

## 13 Bibliografia

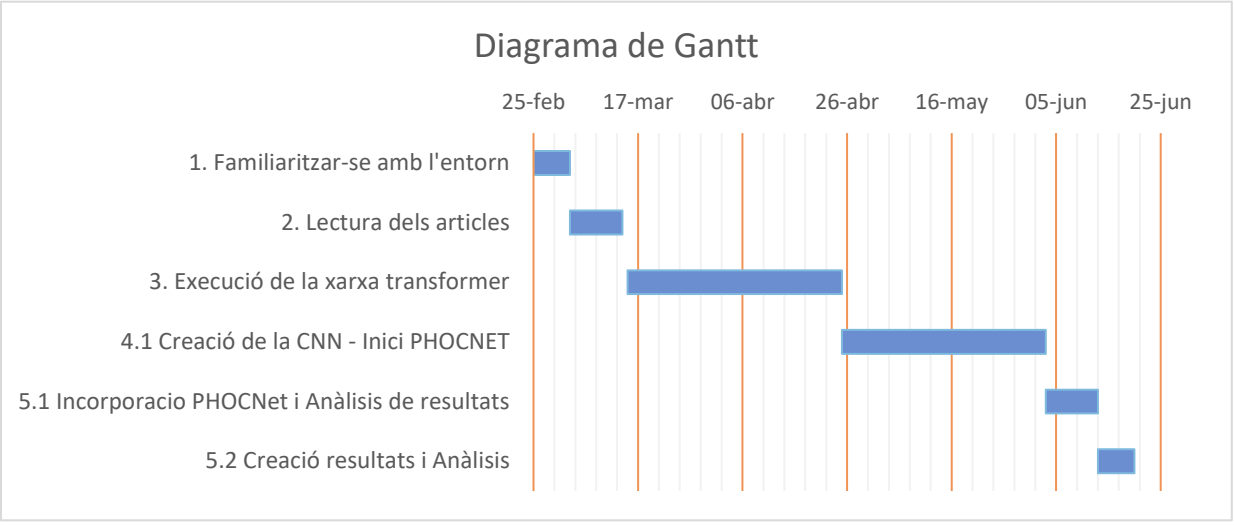
[1] Pytorch. https://pytorch.org
[2] PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. 1604.00187v3. 2017
[3] Bidirectional Scene Text Recognition with a Single Decoder. 1912.03656v2. 2020
[4] Tensorflow install. https://www.tensorflow.org/install
[5] Torch install. https://github.com/pytorch/pytorch
[6] Torchvision install. https://github.com/pytorch/vision
[7] Seaborn. https://github.com/pytorch/vision
[8] Tutorial de pytorch. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
[9] Python: https://www.python.org
[10] K. He et al. Deep residual learning for image recognition. In CVPR, pages 770–778, 2016.
[11] TensordboardX: https://www.tensorflow.org/tensorboard
[12] Numpy: https://numpy.org
[13] PHOCNet pytorch : https://github.com/sghoshcvc/pytorch-phocnet
[14] M. Jaderberg et al. Deep features for text spotting. In ECCV, 2014.
[15] IAM Handwriting Database: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database
[16] S. M. Lucas et al. ICDAR 2003 robust reading competitions. In ICDAR, pages 682–687, 2003.
[17] D. Karatzas et al. ICDAR 2013 robust reading competition. In ICDAR, pages 1484–1493, 2013.
[18] D. Karatzas et al. ICDAR 2015 competition on robust reading. In ICDAR, pages 1156–1160, 2015.
[19] K. Wang et al. End-to-end scene text recognition. In ICCV, pages 1457–1464. IEEE, 2011.
[20] T. Quy Phan et al. Recognizing text with perspective distortion in natural scenes. In ICCV, pages 569–576, 2013.
[21] T. Quy Phan et al. Recognizing text with perspective distortion in natural scenes. In ICCV, pages 569–576, 2013.
[22] A. Mishra et al. Scene text recognition using higher order Language priors. In BMVC-British Machine Vision Conference.
[23] A. Risnumawan et al. A robust arbitrary text detection System for natural scene images. Expert Systems with Applications, 41 (18):8027–8048, 2014.
[24] Codi original Bi-STET: https://github.com/MauritsBleeker/Bi-STET
[25] PHOCNet: https://github.com/ssudholt/phocnet
[26] Cmake: https://cmake.org/download/
[27] Ninja: https://ninja-build.org/
[28] M. D. Zeiler. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701, 2012.

| *Method* | IIIT5K | SVT | IC03 | IC13 | IC15 | SVTP | CUTE |
|---|---|---|---|---|---|---|---|
| O *Left to right* | 94.2 | 88.3 | 93.8 | 85.1 | 75.2 | 78.8 | 81.8 |
| O *Right to left* | 94.1 | 87.9 | 94.8 | 85.7 | 73.3 | 79.5 | 83.6 |
| O *Bidirectional* | 94.7 | 89.0 | 95.2 | 85.8 | 75.8 | 80.6 | 82.5 |
| *Left to right* | 76.1 | 73.9 | 86.5 | 72.4 | 54.3 | 59.8 | 56.6 |
| *Right to left* | 74.8 | 73.0 | 86.9 | 72.5 | 52.8 | 57.4 | 55.6 |
| *Bidirectional* | 76.9 | 75.4 | 87.8 | 73.5 | 55.3 | 61.7 | 57.6 |

*Il·lustració 3: Resultats originals pels datasets, resultats per al dataset reduït amb només 100k iteracions*

# APÈNDIX

## A1. Diagrama de Gantt




## A2. Learning rate 0'1 amb reducció 100k iteracions

| | *IIIT5K* | *SVT* | *IC03* | *IC13* | *IC15* | *SVTP* | *CUTE* |
|---|---|---|---|---|---|---|---|
| *l-t-r* | 70.4 | 69.2 | 82.1 | 67.7 | 50.2 | 56.7 | 47.2 |
| *r-t-l* | 69.6 | 67.5 | 82.4 | 67.6 | 49.3 | 52.9 | 49.3 |
| *bid* | 71.4 | 70.0 | 83.5 | 68.9 | 50.3 | 68.4 | 50.3 |