



XNAP-TRANSLATION GROUP

Introducció i objectius

Aquest projecte està enfocat a la creació d'un model de màquina automàtica per a la traducció de diferents llengües com seria passar de l'anglès al català o a l'espanyol. En aquest projecte crearem un RNN sequence to sequence a Keras per traduir d'una llengua a una altra. El principal objectiu d'aquest treball és entendre el funcionament i l'estructura interna d'aquest model per poder fer-ne les modificacions pertinents per arribar a obtenir un millor model, és a dir, que sigui capaç de fer bones traduccions donades unes paraules o frases.

Code structure

El nostre projecte ha agafat com a punt de partida el codi donat el qual es tractava de 3 arxius principals: el `training.py`, el `util.py` i el `predictionTranslation.py`. L'arxiu principal que crea el model és el `training.py`. En aquest arxiu s'importa tota la informació de l'arxiu `util.py`, definim principalment algunes de les variables que volem que tingui el nostre model i també és on creem i entrenem el nostre model. En `util.py` és s'ón tenim la majoria de les funcions i també definim variables com serien el learning rate, l'optimizer o latent dimension que anirem editant per tal d'aconseguir un millor resultat. Finalment, en l'arxiu `predictionTranslation.py` és on a partir del model creat i d'una paraula o frase en fem la predicció de la seva traducció.

How to use it

1. Descarrega el training dataset: Anglès - Castellà: carpeta spa-eng Anglès - Català: carpeta cat-eng [Altres idiomes] (<http://www.manythings.org/anki/>)
2. `python3 training.py`
3. `python3 predictionTranslation.py`

Dataloader

En aquest projecte tractem principalment la traducció de l'anglès a l'espanyol, tot i que també provem la traducció a altres idiomes com el català. Les dades les tenim en el mateix format, un fitxer per parella d'idiomes on cada línia té la paraula o conjunt de paraules en un idioma (del qual volem la traducció) i després en l'altre (la traducció).

El primer problema amb el qual ens trobem és la quantitat de les dades. En català tenim 1.336 dades les quals són relativament poques.

D'altra banda, tenim l'arxiu en espanyol que té gairebé 140.000 (139.705) però no les podem agafar totes les dades de cop, ja que no era viable processar-les a la GPU alhora. Vam estar buscant altres maneres de gestionar-les sense la necessitat de processar-les totes a la vegada, al principi vam fer un bucle on anava agafant-les de 30.000 en 30.000 i guardàvem els pesos cada cop que acabava el bucle perquè en la següent iteració no s'inicialitzessin a 0 sinó que agafes els de la iteració anterior. Aquesta opció ens ha comportat

algunes dificultats i problemes i, per tant, hem optat per canviar-ho i hem acabat fent un dataloader per agafar les dades, hem pogut arribar a agafar 90.000 de les 140.000 (un 65% de les dades).

Aquesta funció es troba al codi en l'arxiu util.py i s'anomena create_data_loader().

Arquitectura

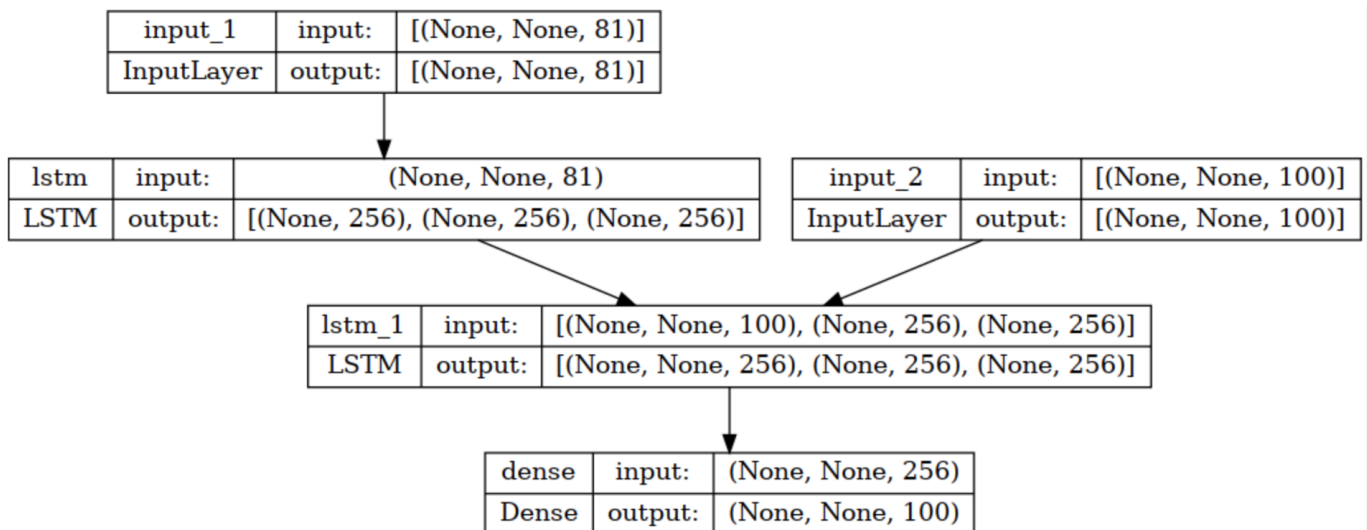
Tractem amb models sequence to sequence (Seq2seq) que converteixen seqüències d'un domini a un altre, com seria en el nostre cas de l'anglès al català/espanyol. Són combinacions de dos RNN, un fa d'encoder i l'altre de decoder. En aquest cas, utilitzem LSTM i GRU, que són dos tipus de RNN.

Model LSTM

Passem la seqüència d'entrada, que ha estat codificada amb one hot encoding, per l'encoder. Té mida 81, que és el nombre de caràcters diferents de la llengua d'entrada, és a dir, de l'anglès. L'encoder processa la seqüència d'entrada i retorna el seu estat intern.

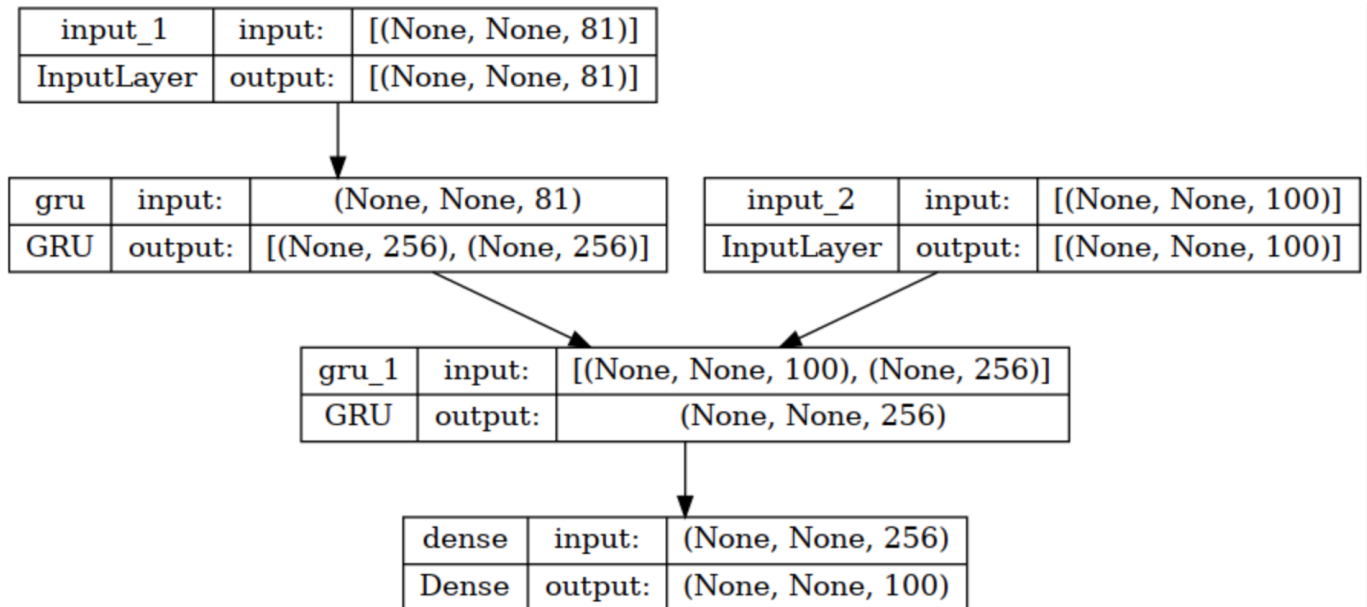
L'input del decoder (segona capa lstm) és la seqüència de sortida, amb mida 100, ja que són els caràcters únics de la llengua de sortida, el castellà. El decoder s'entrena per predir el següent caràcter de la seqüència target, s'entrena perquè produeixi la mateixa seqüència, però un pas més avançat en el futur, això és un mètode d'aprenentatge anomenat ther forcing. Utilitza com a estat inicial els vectors d'estat del encoder, és la manera que el decoder obtingui informació sobre què ha de generar. Per tant, el decoder aprèn a generar targets $[t+1...]$ quan li passem target $[...t]$, condicionat per la seqüència d'entrada.

L'última capa Dense, és la capa final responsable de mapejar la representació hidden del decoder a l'espai del vocabulari, és a dir passa de 256 (latent dim) a 100, mida del vocabulari del target (castellà).



Model GRU

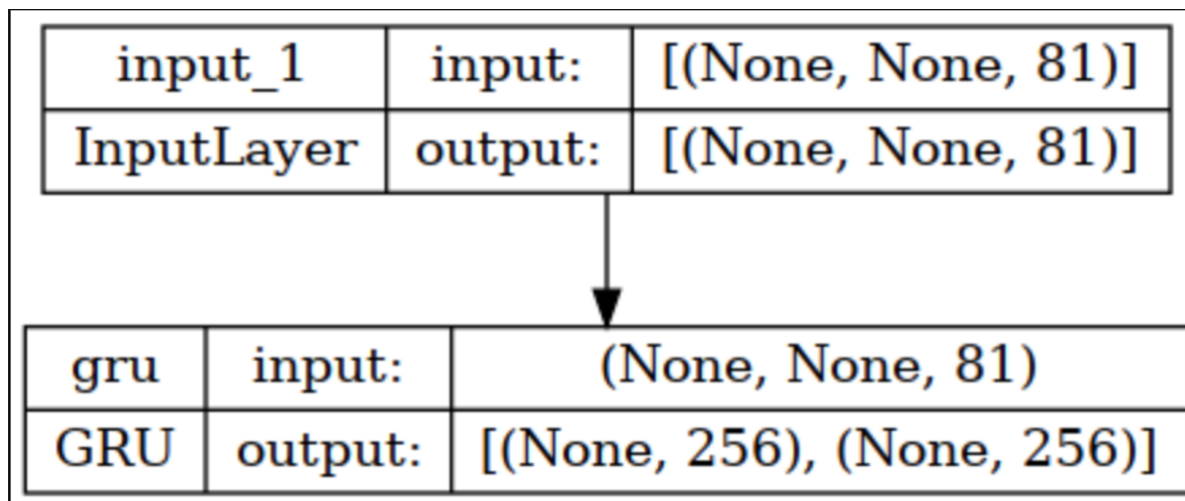
El mateix procés però amb capes GRU:



Aquests models són per entrenar amb dades que ja coneix però per a traduir una seqüència d'entrada desconeguda, hem de fer els model d'inferència.

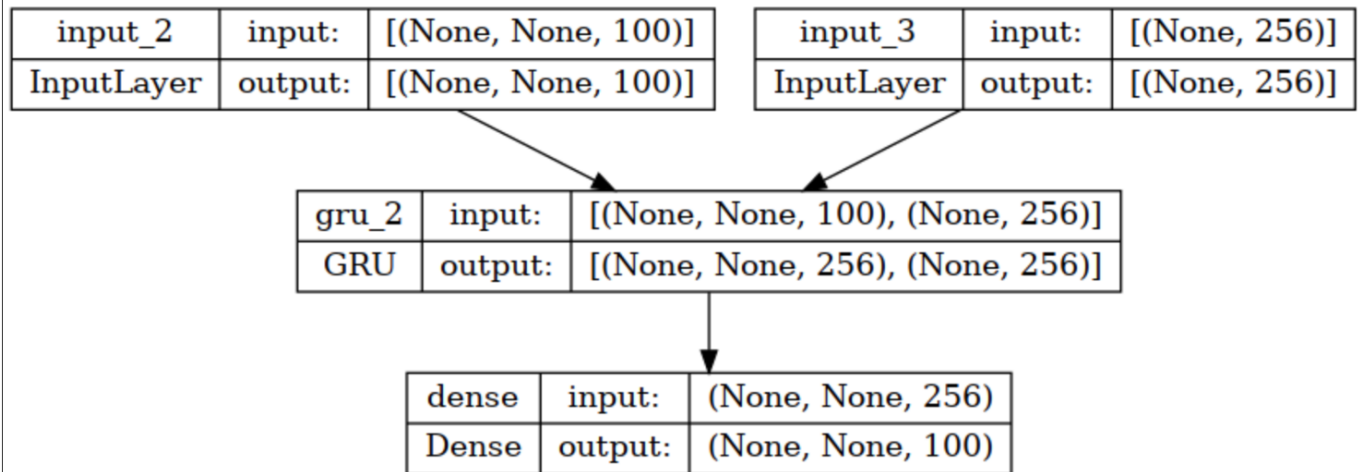
Encoder model inference GRU

A l'encoder li passem la seqüència d'entrada (amb la llengua input anglès) i genera els estats.



Decoder model inference GRU

Al decoder li passem els vectors d'estats i la seqüència de target, per tant, amb dimensió 100, perquè produeixi prediccions pel següent caràcter i després agafem el que tingui màxima predicció. Afegim el caràcter triat a la seqüència de sortida i repetim el procés fins que el caràcter predit és un caràcter especial que marca final de seqüència.



Hiperparàmetres

Per a poder comprovar quins hiperparàmetres són els òptims per al model, s'ha estudiat a partir de l'accuracy i la loss, tant del train com del validation, diferents valors per als paràmetres que es mostren a continuació. El cas base a partir del qual es van modificant els valors del paràmetre estudiat són els que s'ha trobat que són més adients a la teoria. Són, per tant:

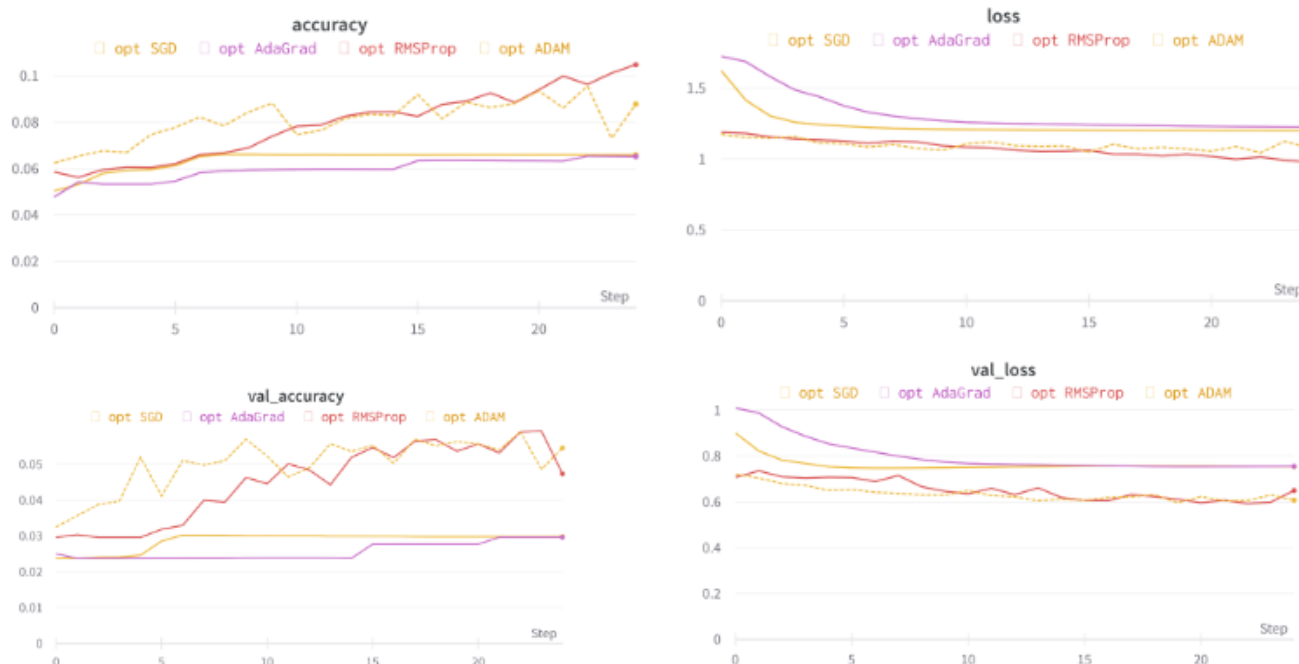
- Epochs:25
- Optimizer: RMSProp
- Learning rate: 0,0001
- Sense Dropout
- Cell type: LSTM

Estudi del valor que fa una major accuracy:

Optimizer

Un optimitzador és un algoritme que s'utilitza per ajustar els paràmetres d'un model amb l'objectiu de minimitzar una funció de pèrdua i permeten que els models aprenguin de les dades i millorin el seu rendiment. Alguns optimitzadors poden ser més ràpids que altres en trobar el mínim de la funció de pèrdua. Altres poden ser més robustos als mínims locals o als punts de sella. A més, alguns optimitzadors poden ser més adequats per a certs tipus de problemes o models. És per aquesta raó que s'han provat amb 4 valors d'optimitzer diferents: SGD, AdaGrad, RMSProp i Adam.

- SGD (Stochastic Gradient Descent): És un dels algoritmes més populars per realitzar l'optimització i és el mètode més comú per optimitzar les xarxes neuronals. Actualitza els paràmetres en la direcció oposada del gradient de la funció objectiu respecte als paràmetres.
- AdaGrad: adapta la taxa d'aprenentatge als paràmetres, realitzant actualitzacions més grans per als paràmetres infreqüents i actualitzacions més petites per als freqüents.
- RMSProp: utilitza una mitjana mòbil del quadrat del gradient per normalitzar el gradient.
- Adam: Combina RMSProp i el moment emmagatzemant tant la taxa d'aprenentatge individual de RMSProp com el promig ponderat del moment.



Learning rate

El learning rate fa referència a l'hiperparàmetre que controla l'ajust dels paràmetres del model en resposta a l'error estimat. Determina la mida dels passos en la direcció oposada del gradient durant l'optimització. Un LR alt pot fer que el model convergeixi ràpidament, però també pot fer que salti sobre el mínim i no convergeixi. Una LR baix pot fer que el model convergeixi més lentament, però pot augmentar la precisió de la solució. S'ha provat amb valors diferents perquè es mostrés clarament quin era el valor més adequat. Han estat: 0.1, 0.01 i 0.001.



Drop out

El dropout és un hiperparàmetre que permet prevenir el sobreajust en el model. Consisteix en desactivar aleatòriament algunes unitats de la xarxa durant l'entrenament. Això fa que la xarxa sigui més robusta i menys propensa a memoritzar les dades de train.

A partir de les gràfiques podem veure que és bastant irregular, però la tendència és que el valor de dropout 0 és el que proporciona un accuracy més elevat, de 0.1 aproximadament i, per tant, té una loss més baixa a les èpoques finals, concretament de 1.04.



Cell type

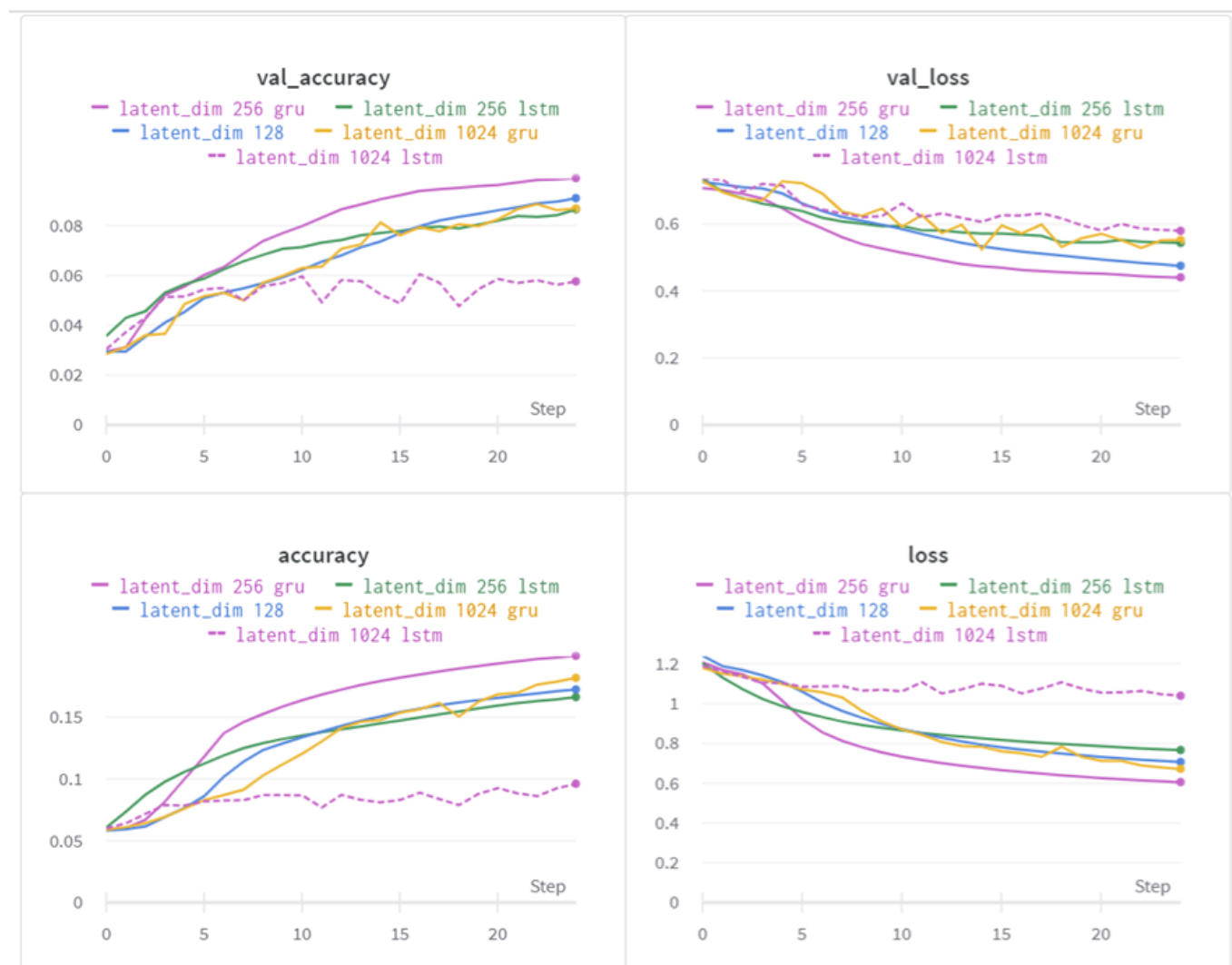
GRU (Gated Recurrent Unit) i LSTM (Long Short-Term Memory) són dos tipus de cel·les recurrents utilitzades en les RNN. Les cel·les GRU i LSTM tenen portes que controlen el flux d'informació a través de la cel·la. Això els permet aprendre dependències a llarg termini en les dades. La principal diferència entre aquest dos tipus de cel·les és que les GRU tenen menys portes i són més simples.

A l'observar les gràfiques tant d'accuracy com de loss, podem veure com aquestes dues s'inicien al mateix punt però ràpidament es diferencien. Per una banda, la GRU (en groc) augmenta ràpidament al llarg de les epochs, mentre que LSTM es manté més constant al llarg de l'entrenament en un valor més baix en accuracy i major en loss.

Latent dimension

El latent_dim representa el nombre de cel·les de memòria o unitats que hi ha a la capa LSTM o GRU. Cada una d'aquestes cel·les reté informació amb el temps i interactua amb altres cel·les de memòria. Hem provat 3 valors diferents 128, 256 i 1024. En la gràfica observem que el millor resultat és obtingut pel 256. Això ho explica el fet que tant el 128 com el 1024 poden crear una tendència a underfitting i overfitting

respectivament. Un agafant massa poca informació i tornant-se així més simple i l'altre per la contra agafant massa informació i fent-se més complex.



Mètriques

Respecte a les mètriques com hem dit anteriorment, hem utilitzat l'accuracy per efectuar totes les modificacions d'hiperparàmetres, ja que es tracta d'una mètrica senzilla i fàcil de veure com de bé funciona el nostre model canviant els hiperparàmetres, sense necessitat d'aprofundir molt. Aquesta mètrica té els seus inconvenients com seria no tenir en compte el context, la fluïdesa o la coherència de la traducció que es fa. Per això hem implementat també la mètrica BLEU (Bilingual Evaluation Understudy).

Obtenim que de la paraula d'entrada en anglès és 'What?' obtenim en espanyol '¿Quu', la traducció correcta seria '¿Qué?'. No és òptima però per tenir un 0.25 d'accuracy, que és el que ens dona el nostre millor model està força bé. Hi ha diferents mètodes objectius:

- Accuracy: és el més simple i compta les coincidències exactes entre la traducció generada pel model i la de referència. D'aquesta manera proporciona una idea general de com que bé o no prediu el model en comparació a les traduccions que s'utilitzen com a referència. És per això que no té en compte altres aspectes com el context, coherència o fluïdesa de les traduccions.
- WER (Word Error Rate): taxa d'error de paraules, és una mètrica que compara la sortida generada pel sistema de traducció automàtica amb una referència. Es calcula comptant el nombre total de

paraules errònies (eliminacions, insercions i n) i dividint-lo pel nombre total de paraules a la referència. El resultat s'expressa com un percentatge d'errors.

- PER (Position-Independent Word Error Rate): taxa d'error de paraules independent de la posició, és similar al WER però no té en compte la posició específica dels errors. Això significa que es considera un error si una paraula s'omet, s'insereix o es substitueix, sense importar la seva posició a la frase. Es calcula dividint el nombre total d'errors pel nombre total de paraules a la referència.
- mWER (Multi-Reference Word Error Rate): taxa d'error de paraules amb múltiples referències, s'utilitza quan hi ha múltiples traduccions de referència disponibles. En lloc de comparar la sortida del sistema de traducció automàtica amb una sola referència, es compara amb totes les referències disponibles. Es calcula trobant la referència que tingui el menor nombre d'errors i després s'aplica la fórmula del WER a aquesta referència específica.
- BLEU (Bilingual Evaluation Understudy): àmpliament utilitzada per avaluar la qualitat de les traduccions automàtiques en comparació amb una o més traduccions de referència. Mesura la similitud entre la traducció generada i les referències basant-se en la coincidència de paraules o frases. Com més gran sigui el valor de BLEU, millor serà la qualitat de la traducció. BLEU considera la precisió unigram, bigram, trigram i quadrigram, i té en compte la brevetat de les traduccions.

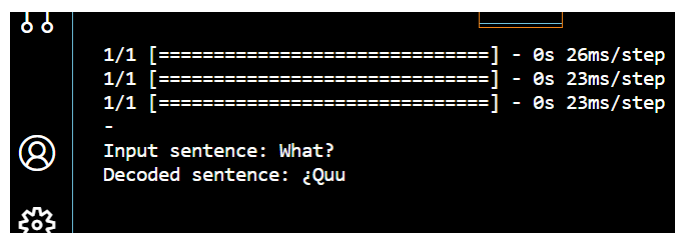
S'ha utilitzat la mètrica accuracy per a fer l'estudi dels hiperparàmetres i del model, però s'ha implementat també la mètrica BLEU score. En aquest cas, el resultat de la mètrica és inferior que a l'aplicar la mètrica accuracy.

Resultats

Podem concloure que els millors hiperparàmetres que ens ha donat han estat els següents:

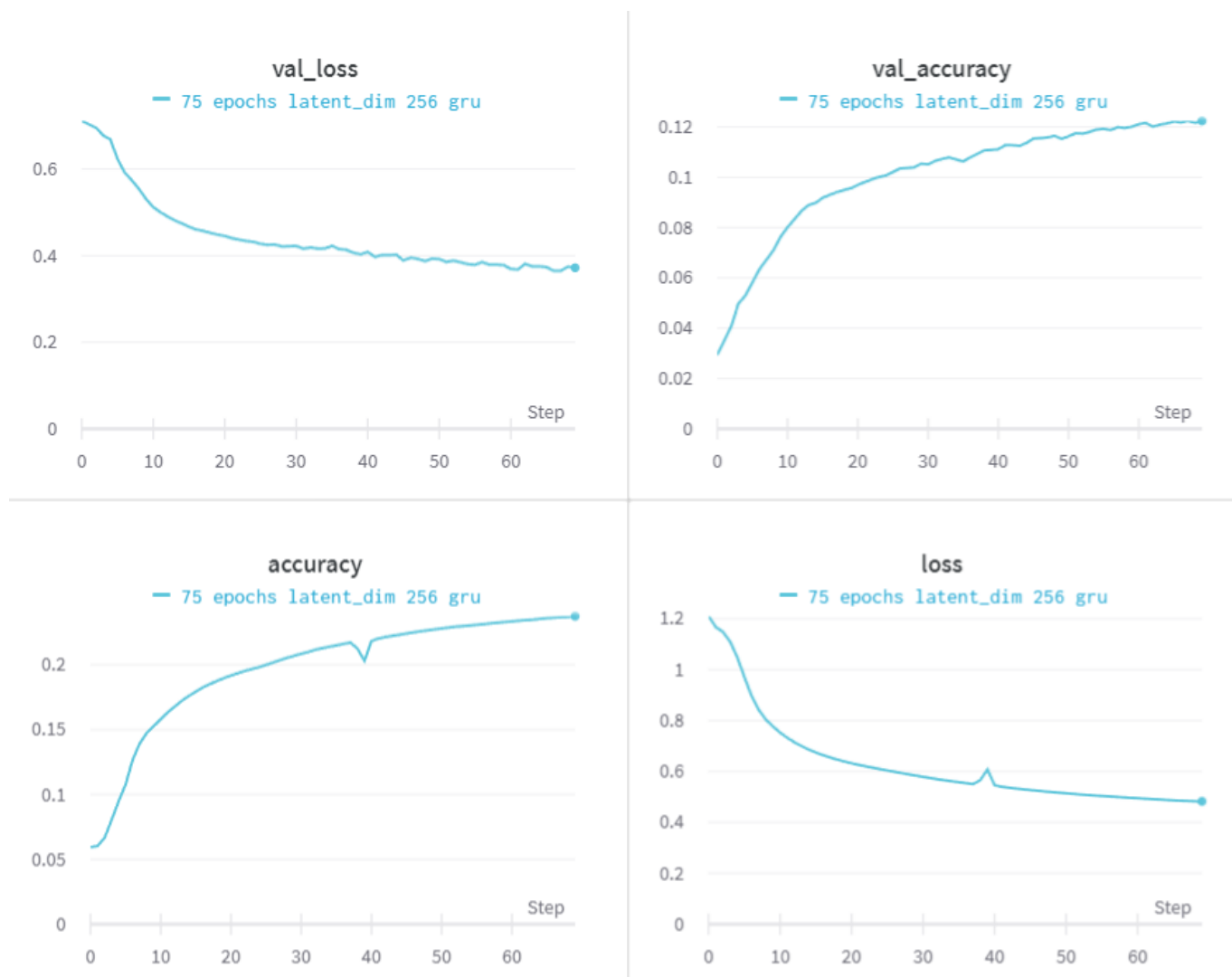
- Epochs:75
- Optimizer: RMSProp
- Learning rate: 0,0001
- Sense Dropout
- Cell type: GRU

En aquesta primera imatge hem executat el predictionTranslation amb el nostre millor model:



```
1/1 [=====] - 0s 26ms/step
1/1 [=====] - 0s 23ms/step
1/1 [=====] - 0s 23ms/step
-
Input sentence: What?
Decoded sentence: ¿Quu
```

Les mètriques resultants d'aquest model són les que es mostren a continuació:



Podem veure com al executar més èpoques amb els paràmetres que donen un resultat més òptim prèviament, el resultat tant de la mètrica accuracy com loss milloren i per tant, arriba a un valor de 0.25 i 0.4 a l'entrenament i 0.12 i 0.37 respectivament a la validació.

Conclusions

Per concloure, després d'haver tingut diverses dificultats com ha sigut la capacitat de dades a processar alhora, les quals si haguéssim pogut n'hauríem agafat més, hi ha millores a fer. Entre d'elles estaria fer més proves d'altres hiperparàmetres tot i que de primeres no creguéssim que són els més òptims i també deixar més èpoques a les execucions. Cal dir que tot i haver provat de traduir d'un idioma a un altre i viceversa podríem haver provat més idiomes.

Contributors

Alba Ballarà 1597532@uab.cat

Berta Carner 1600460@uab.cat

Blanca de Juan 1604365@uab.cat

Xarxes Neuronals i Aprenentatge Profund Grau de Enginyeria de Dades, UAB, 2023