NATURAL LANGUAGE PROCESSING

IMAGE CAPTIONING

NICOLAE DUCAL, ALEXANDRU IOAN ȚIFUI, DANIEL AVRAM
GRUPA 334

05.05.2022

CE REPREZINTĂ PROBLEMA DESCRIERII IMAGINILOR?

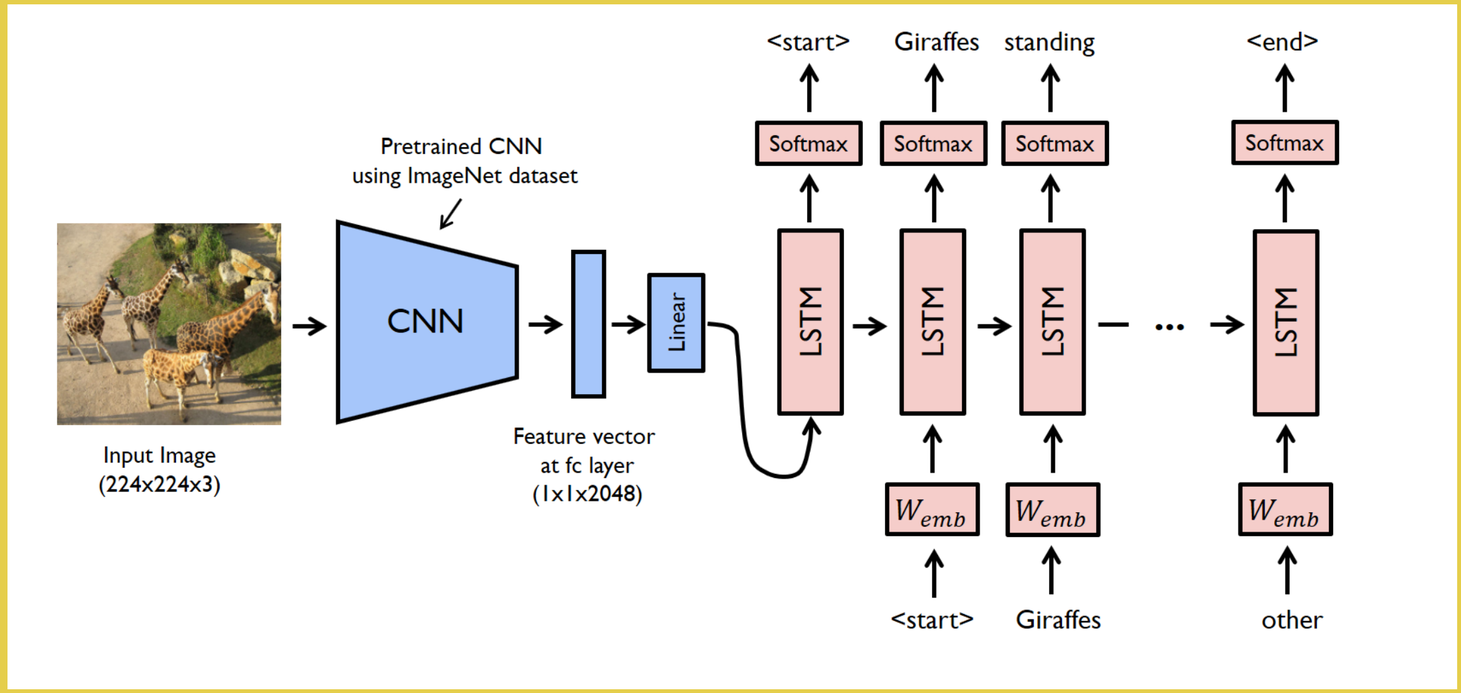
Descrierea imaginilor (sau în engleză **Image Captioning**) reprezintă procesul de generare a unei descrieri a conținutului unei imagini primite ca input.

Această problemă este una des întâlnită în domeniul Inteligenței Artificiale, care a evoluat în ultimii ani destul de mult, oferind rezultate foarte apropiate de descrierile naturale. De asemenea, acest proces îmbină atât domeniul de **Procesare a Limbajului Natural (NLP)**, cât și domeniul de **Vedere Artificială (Computer Vision)**.

"construction worker in orange safety vest is working on road."

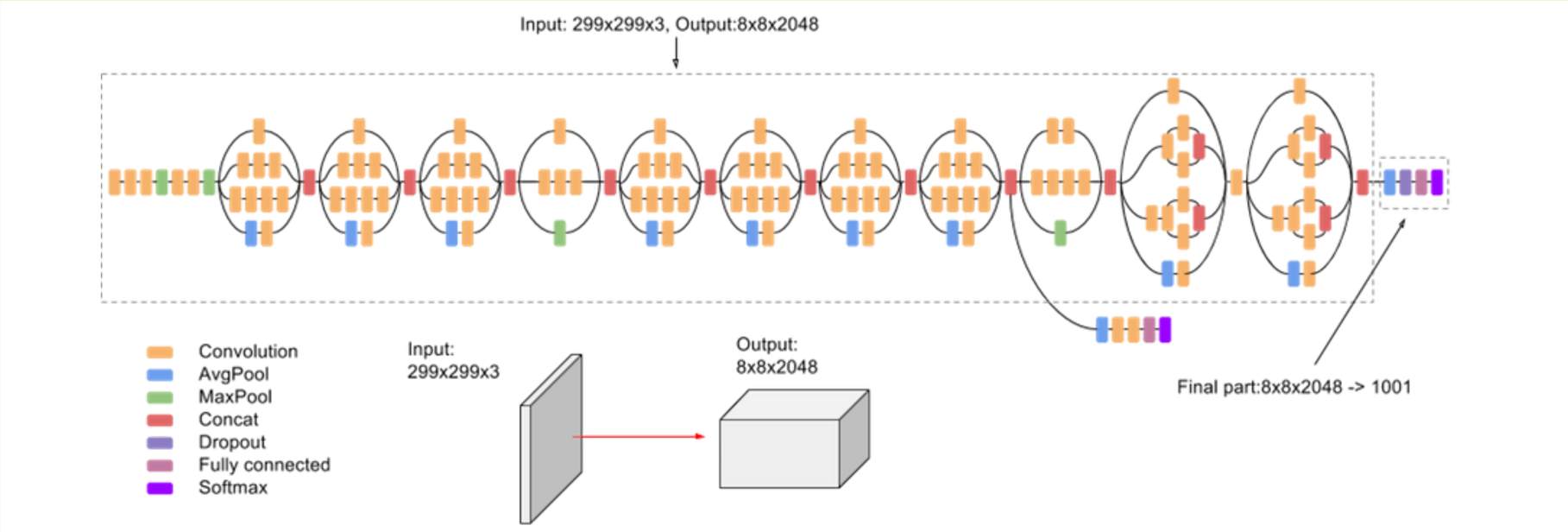
CUM FUNCȚIONEAZĂ DESCRIEREA IMAGINILOR

Există mai multe variante de implementare pentru rezolvarea problemei de descriere a imaginilor, iar metode mai optimizate și mai precise încă mai apar. Implementarea pe care am folosit-o în cadrul proiectului nostru este cea ilustrată și în figura de mai jos, care folosește o **Rețea Neuronală Convoluțională (CNN) preantrenată**, utilizată ca extractor de caracterisici din imagini (**feature extractor**), o **Rețea Neuronală Convoluțională pentru encoding-**ul caracteristicilor extrase (care va conține un singur strat liniar) și o **Rețea Neuronală Recurentă (RNN)**, formată din **LSTM**-uri (**Long Short-Term Memory**).



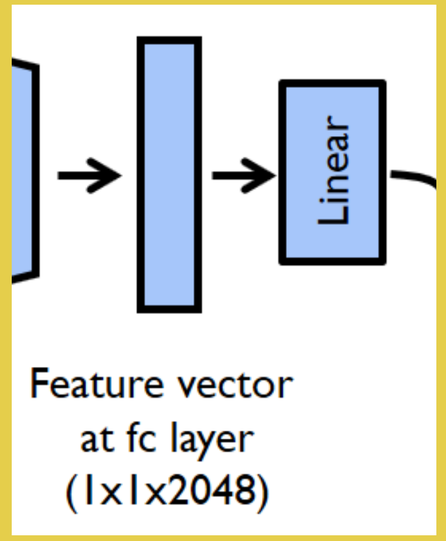
FEATURE EXTRACTOR CNN

Pentru etapa de extragere a caracteristicilor din imagini, am utilizat modelul **Inception-V3**, preantrenat pe datasetul ImageNet din framework-ul Pytorch, din care însă am omis partea de Fully-Connected. În figura de mai jos, putem observa structura arhitecturii: primește ca input imagini color de dimensiune 299x299, conține câteva blocuri de inception, iar ca output, features-urile noastre vor fi de dimensiune 8x8x2048. De asemenea, pentru a scăpa de cele 3 dimensiuni, am decis să trecem de la dimensiunea 8x8x2048 la dimensiunea 64x2048 pentru a putea fi folosite mai ușor la partea de RNN.



ENCODING CNN

După extragerea caracteristicilor unei imagini (de dimensiune 64x2048), trebuie să folosim o Rețea Neuronală Convoluțională, formată dintr-un singur strat liniar, care va transforma caracteristicile din dimensiunea inițială în dimensiunea **64xEmbedding_Size**, unde Embedding_Size reprezintă dimensiunea embedding-ului folosit în cadrul RNN-ului și este setată de utilizator înainte de antrenarea modelului. De asemenea, folosim și un strat de Dropout pentru regularizare și prevenirea overfitting-ului.

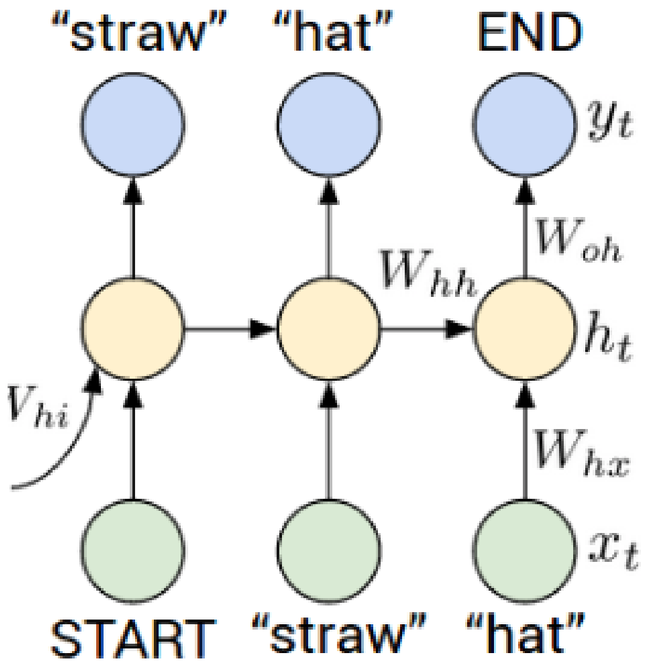



```python
class EncoderCNN(nn.Module):
    def __init__(self, embedding_size):
        super(EncoderCNN, self).__init__()
        self.linear = nn.Linear(in_features=2048, out_features=embedding_size)
        self.activation = nn.ReLU()
        self.dropout = nn.Dropout(0.25)

    def forward(self, x):
        features = self.linear(x)
        return self.dropout(self.activation(features))
```

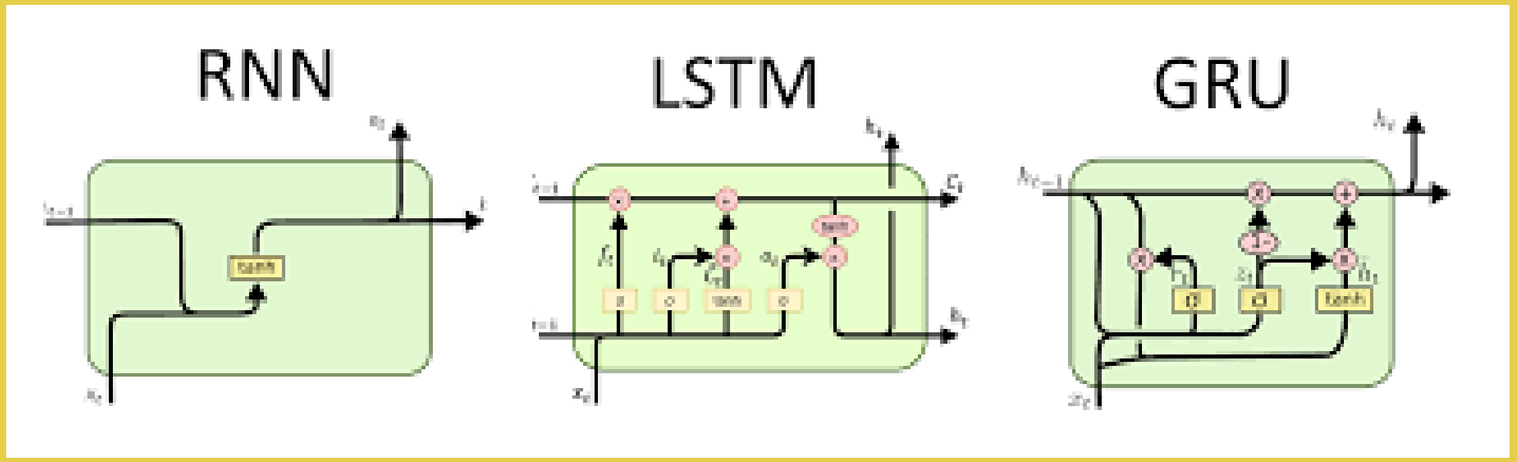
CAPTIONING RECURENT NEURAL NETWORK

CaptioningRNN este o rețea neuronală recurentă compusă din **Embedding** și un **RNN**. În urma encodării features-urilor, la propagarea forward, imaginea trece împreună cu descrierea ce reprezintă ground-truth in cazul antrenării, sau împreună cu cuvintele detectate la pașii precedenți în cazul prezicerilor printr-un **Embedding Layer**, iar ulterior le trece printr-un **RNN**, format din **LSTM**-uri. În cazul antrenării rețeaua returnează un strat fully connected de dimensiunea vocabularului pentru calcularea loss-ului, în cazul evaluării returnează vectorul cu indicii cuvintelor descrierii.



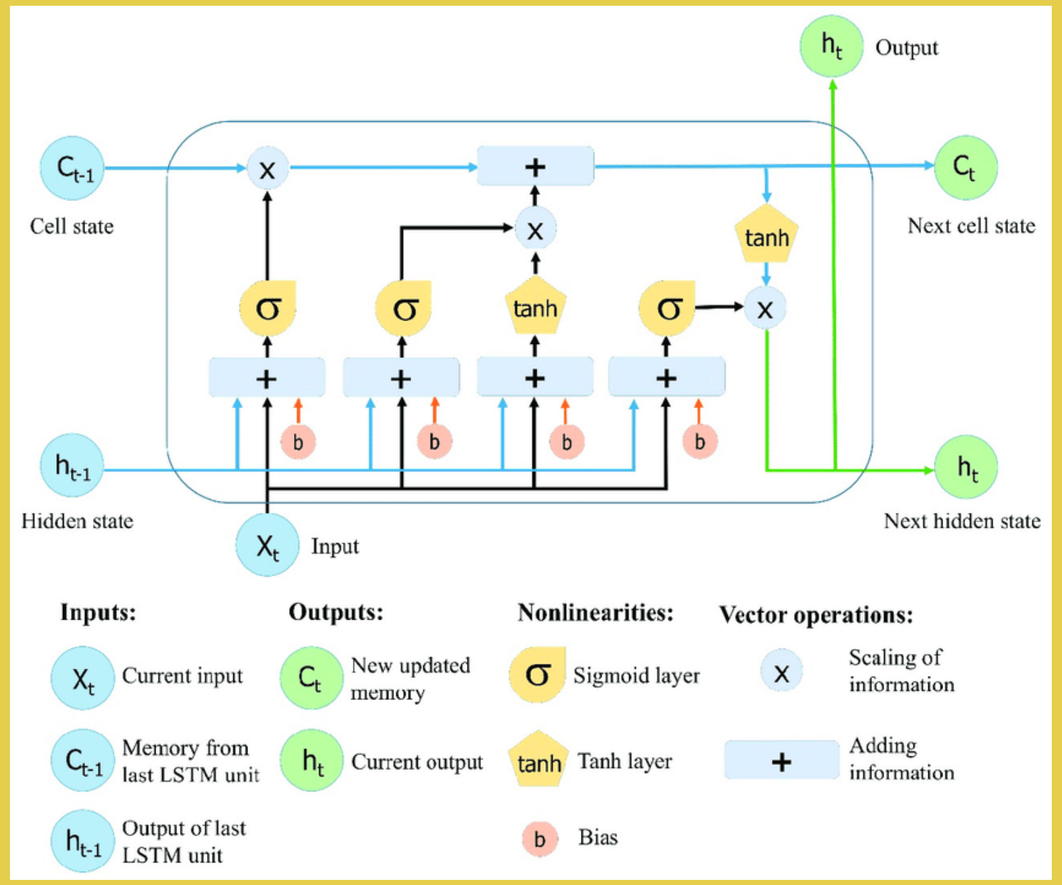
CE ESTE O REȚEA NEURONALĂ RECURENTĂ?

Rețelele Neuronale Recurente sunt un tip de rețele neuronale adaptate pentru detectarea datelor secvențiale. În cazul rețelelor neuronale obișnuite care au scopul de a genera mai multe detecții, în urma unei propagării forward, detecțiile generate la un anumit pas nu depind de detecțiile precedente (de exemplu: în cazul R-CNN, fiecare detecție dintr-o imagine este independentă una față de alta).
În cazul în care detecția curentă depinde de detecțiile precedente, se va folosi RNN. În cazul nostru, evaluarea unei imagini va genera o secvență de cuvinte unde fiecare cuvânt va depinde nu doar de features detectate în imagine, dar și de cuvintele detectate precedent (cu scopul de a lega corect gramatical cuvintele, sau de a nu repeta aceeași idee de două ori).



LONG SHORT-TERM MEMORY

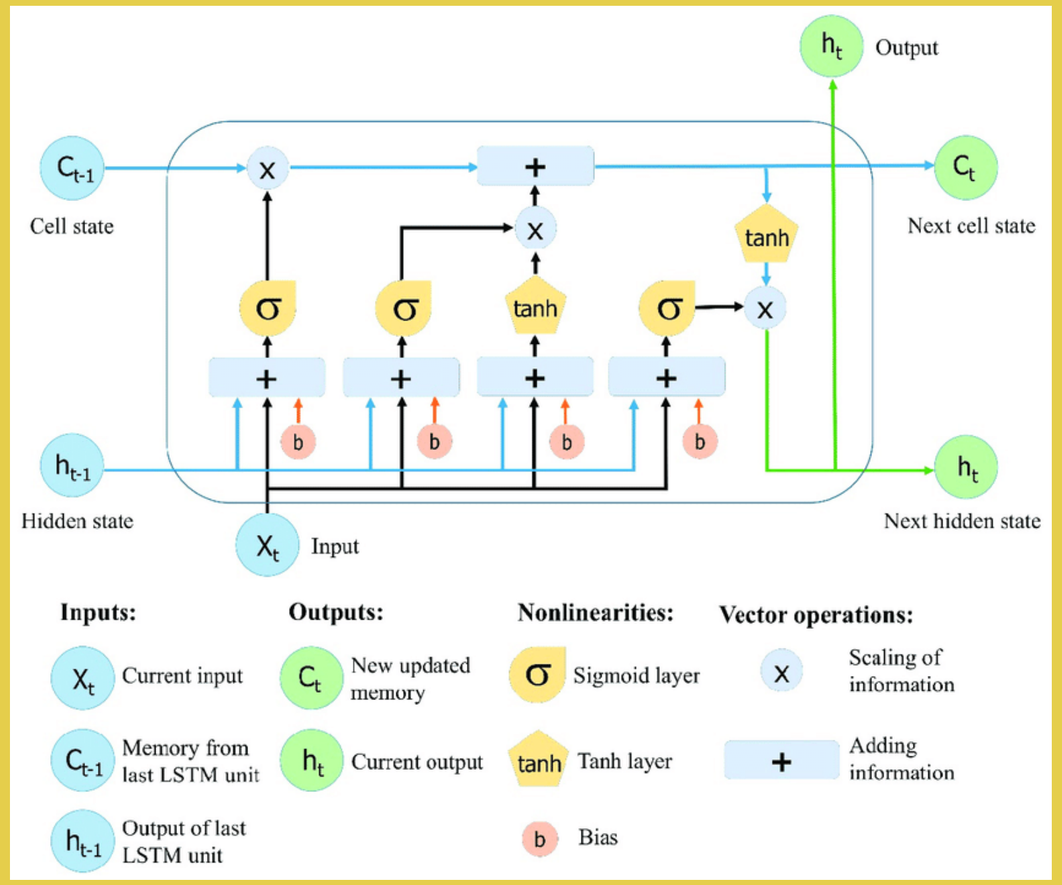
Long Short-Term Memory (LSTM) este un tip de RNN care este capabil să învețe dependențe de lungă durată. Acestea primesc 3 intrări: *Input*-ul **xt**, starea celulei precedente **C(t-1)** și starea precedentă ascunsă **h(t-1)**, și returnează output-ul **ht** și starea celulei curente **Ct** care sunt folosite pentru a compune următorul *Output*.
În cazul problemei de *Image Captioning*, starea primei celule **C1** va fi *encoding*-ul de *features* al imaginii, iar input-ul **xt** va fi al t-lea cuvânt din caption.



LONG SHORT-TERM MEMORY

Long Short-Term Memory (LSTM) este format din 3 porți: **Forget Gate**, **Input Gate** și **Output Gate**.

- **Forget Gate** decide condiționat de **h(t-1)** și **xt** ce informație din **C(t-1)** să fie uitată (în urma adunării, trecerii prin activarea *sigmoid* și înmulțirea cu starea precedentă, valorile înmulțite cu 0 vor fi uitate)
- **Input Gate** crează starea celulei curente. Mai întâi, este creat un vector de valori candidate prin înmulțirea activării *sigmoid* și activării *tanh* a sumei vectorilor **h(t-1)** și **xt**. Apoi, aceste valori candidat sunt adunate cu valorile obținute la **Forget Gate** (cea care decide care valori sunt păstrate din celula precedentă).
- **Output gate** va calcula output-ul celulei curente (și starea ascunsă) prin înmulțirea activării *sigmoid* a sumei dintre **h(t-1)** și **xt** și activarea *tanh* a stării celulei calculate la **Input Gate**.



A black dog and a spotted dog are fighting .
A black dog and a tri-colored dog playing with each other on the road .
A black dog and a white dog with brown spots are staring at each other in the street .
Two dogs of different breeds looking at each other on the road .
Two dogs on pavement moving toward each other .

DATASETUL

Pentru antrenarea modelului am folosit setul de date **Flickr8k** de pe kaggle. Acest dataset servește drept benchmark pentru modele de Image Captioning și conține 8000 de imagini cu câte 5 descrieri fiecare.
În urma lemmatizării, trecerii printr-un *freq_threshold = 5* și adăugarea tokenilor ***<PAD>***, ***<SOS>***, ***<EOS>*** și ***<UNK>***, vocabularul cuvintelor va conține 2994 de tokeni.

ANTRENAREA MODELULUI

La antrenare, imaginile, împreună cu descrierile lor adevărate vor fi trecute prin modelul nostru ce îmbină cele 3 părți descrise anterior. Ca **funcție de loss** folosim **CrossEntropyLoss**, iar ca **optimizator - Adam** cu **Learning Rate 0.0005**. Feature Extractor-ul nu se va antrena în urma propagării backwards, iar *LSTM*-urile vor primi ca input **x** cuvintele din ground truths.

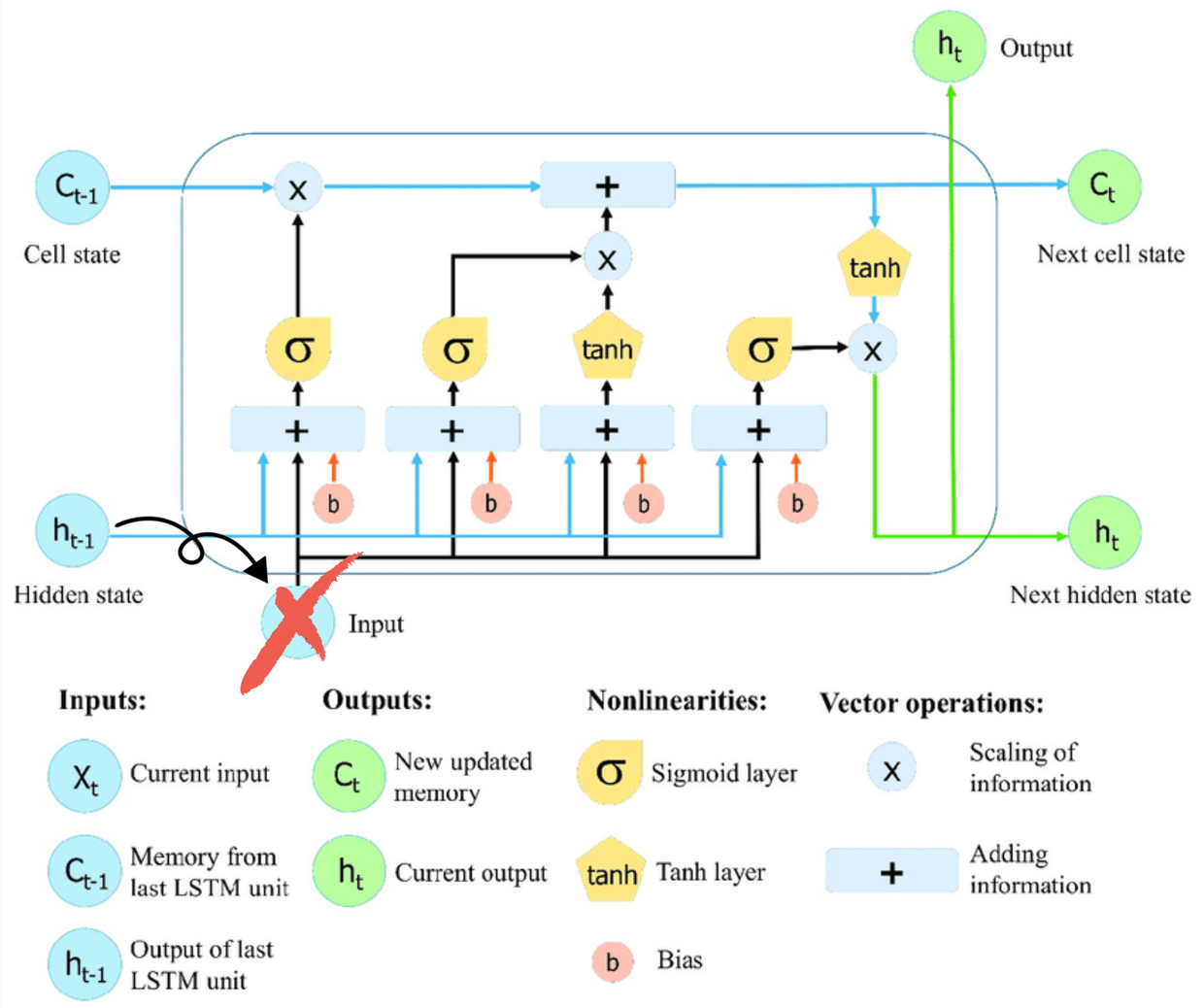
OPTIMIZAREA ANTRENĂRII

Una din problemele întâlnite în procesul de antrenare utilizat inițial a fost extragerea caracteristicilor din imaginile din setul de antrenare la fiecare epocă. Modelul de InceptionV3 este unul destul de complex, care efectuează multe operații pentru a extrage features dintr-o singură imagine, care chiar și cu GPU, acest timp este destul de neneglijabil. Din acest motiv, abordarea inițială lua cel puțin 24 minute per epocă pe platforma Google Colab. Pentru a antrena cel puțin 100 de epoci ar fi trebuit să așteptăm ~2400 minute, sau 40 de ore, ceea ce nu este rezonabil pentru noi.

Pentru a o soluționa, am făcut o mică modificarea în abordarea inițială: în loc să extragem features din imagini în cadrul fiecărei epoci, facem o preprocesare înainte de a antrena și extragem atunci features din fiecare imagine, din cauză că noi nu antrenăm modelul de InceptionV3, ci îl folosim doar pentru a extrage caractersitici din imagini, care vor fi acealeași la fiecare epocă. Apoi, antrenăm modelul folosind aceste caracteristici din care am creat un Dataset și Dataloader diferit. Astfel, am optimizat timpul de antrenare per epocă de la 24 minute la 30 secunde, reducând esențial timpul de antrenare și dându-ne posibilitatea de a antrena un număr mare de epoci pentru rezultate mai bune la etapa de testare.

TESTAREA MODELULUI

La testare, rețele *LSTM* funcționează într-un mod diferit: Din moment ce nu sunt ground truths pentru input-urile **xt**, acestea sunt înlocuite cu output-ul celulei precedente.



REZULTATE

A woman is riding a horse in a field

```
a young girl is riding a horse in a field
```

A boy in blue T-shirt is surfing in the sea

```
a man in a blue wetsuit is wakeboarding in a blue shirt
```

REZULTATE

Two wolves are roaring while another wolf is watching

two dogs are running through the snow

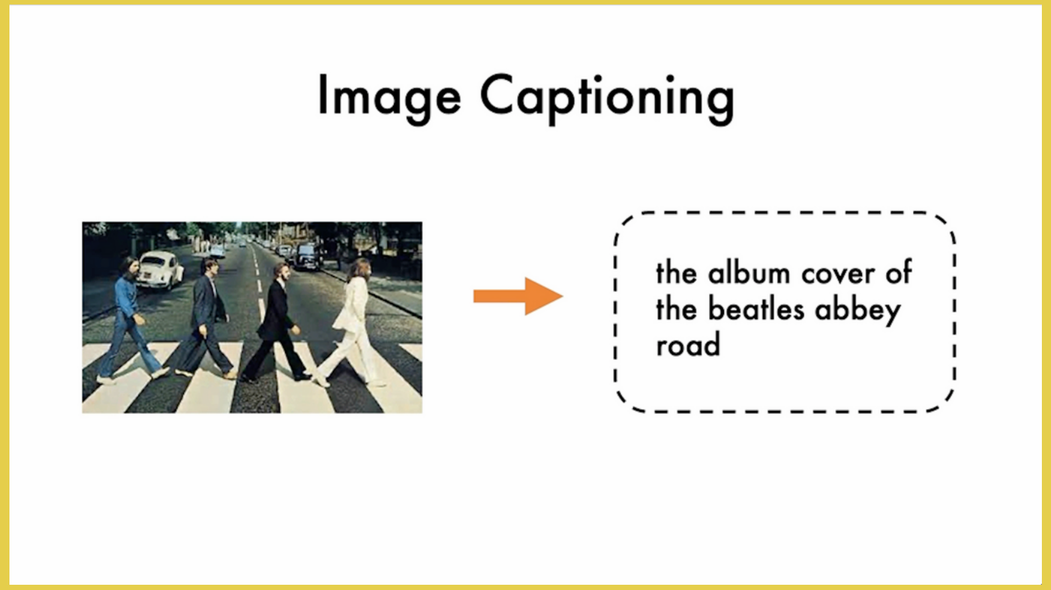
A group of people are running in a marathon

a group of people are standing on a <UNK> watching a <UNK>

ALTE IMPLEMENTĂRI MAI PERFORMANTE

Modelul prezentat și implementat de noi în cadrul acestui proiect este unul destul de direct și simplu de implementat și antrenat, care obține rezultate destul de bune și aproape de adevăr. Însă, pe parcurs, au apărut modele și mai performante care rezolvă problema de image captioning mult mai eficient, mai optim și cu descrieri mult mai vaste și bune. De exemplu, printre aceste modele se numără: **RDN**, **LEMON**, **Oscar**, **OFA**.

Rezultatele pentru cel din urmă (OFA), testat pe poza de copertă a albumului formației Beatles este prezentat în imaginea de mai jos:



REFERINȚE

1. InceptionV3 - https://cloud.google.com/tpu/docs/inception-v3-advanced
2. Image Captioning Tensorflow - https://www.tensorflow.org/tutorials/text/image_captioning
3. Image Captioning - https://towardsdatascience.com/image-captioning-in-deep-learning-9cd23fb4d8d2
4. Image Captioning - https://paperswithcode.com/task/image-captioning
5. OFA - https://paperswithcode.com/paper/unifying-architectures-tasks-and-modalities

MULȚUMIM PENTRU ATENȚIE!