Vision transformer
Proiect IAVA

Sociu Daniel

Ce este un transformer?

- Un transformer este un model de învățare adâncă care se bazează pe mecanismul de self-attention folosit în aplicații de sequence to sequence (encoder - decoder), diferit de rețelele neuronale recursive acesta este capabil de a procesa datele de intrare (secvența) în paralel.
- Este folosit cel mai des în aplicații de procesare de limbaj natural (traducerea unui text, NER (Named Entity Recognition)), dar recent este folosit și în aplicații de vedere artificială

Structura unui transformer

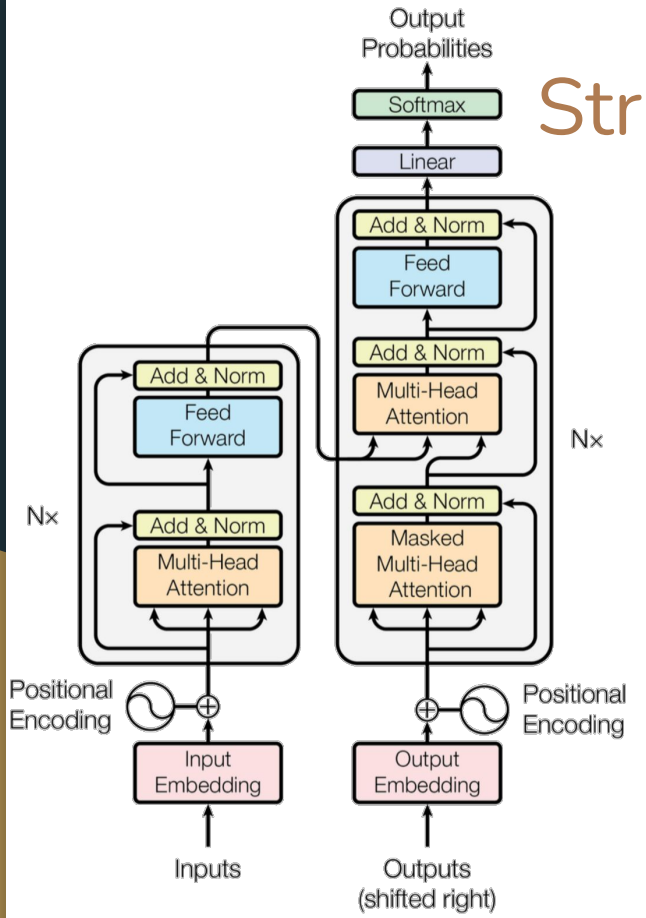


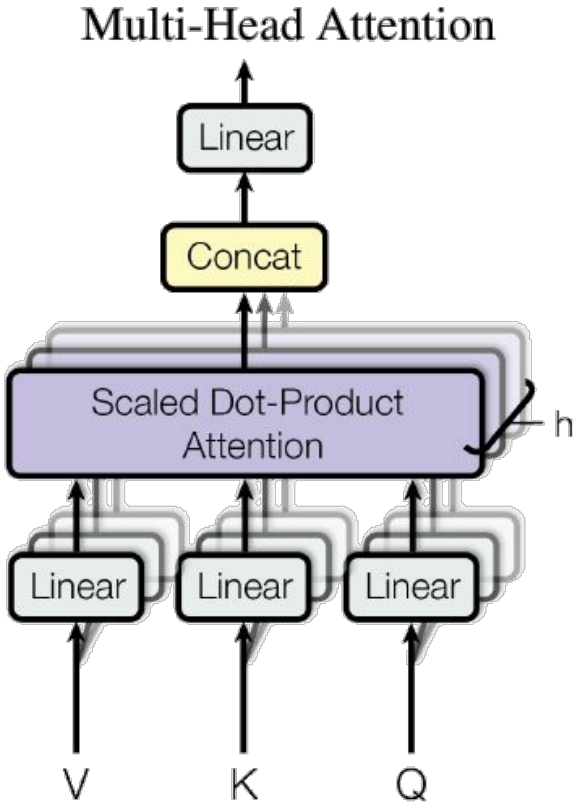
Figure 1: The Transformer - model architecture.

- Este format din două părți:
  - Encoder - conține 2 sub-layers
  - Decoder - conține 3 sub-layers
- De obicei este folosit doar encoder-ul, iar pe output-ul acestuia se adaugă clasificatori
- Sunt foarte importante și legăturile reziduale dintre sub-layere

Intrarea unui transformer

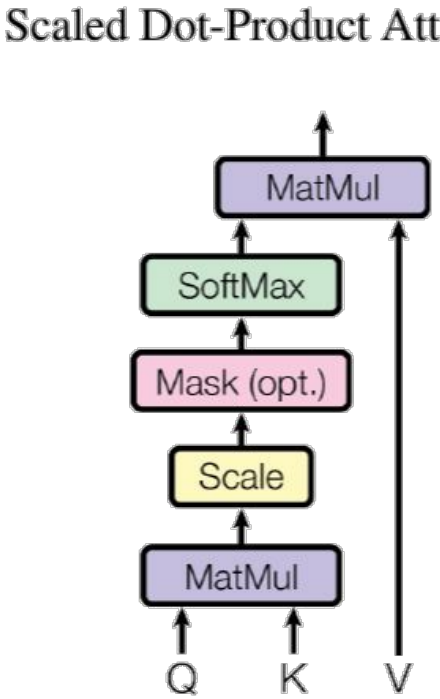
- Intrarea este o secvență de o lungime maximă (teoretică) (e.g. BERT - 512 tokeni) iar acești tokeni este trecut printr-un layer de embedding care transforma tokenii în vectori de lungime fixă
- Deoarece secvența este calculată în paralel se pierde noțiunea de ordine din secvență, de aceea se adaugă stratul de positional encoding
- Intrarea în blocul encoder este de forma (N, d) unde N este lungimea secvenței iar d este lungimea embedding-ului

Multi-head attention (self-attention)



- Primește ca input un querry (Q), chei (K) care corespund unor valori (V), acestea sunt obținute în general prin 3 straturi liniare din embedding-uri
- Vectorii V, K, Q sunt împărțiți în h ”head-uri” care sunt la final combinate la loc în stratul Concat

Scaled Dot-Product Attention



- Acest layer reprezintă atenția transformer-ului
- Diagrama din stânga este reprezentată de următoarea formulă:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

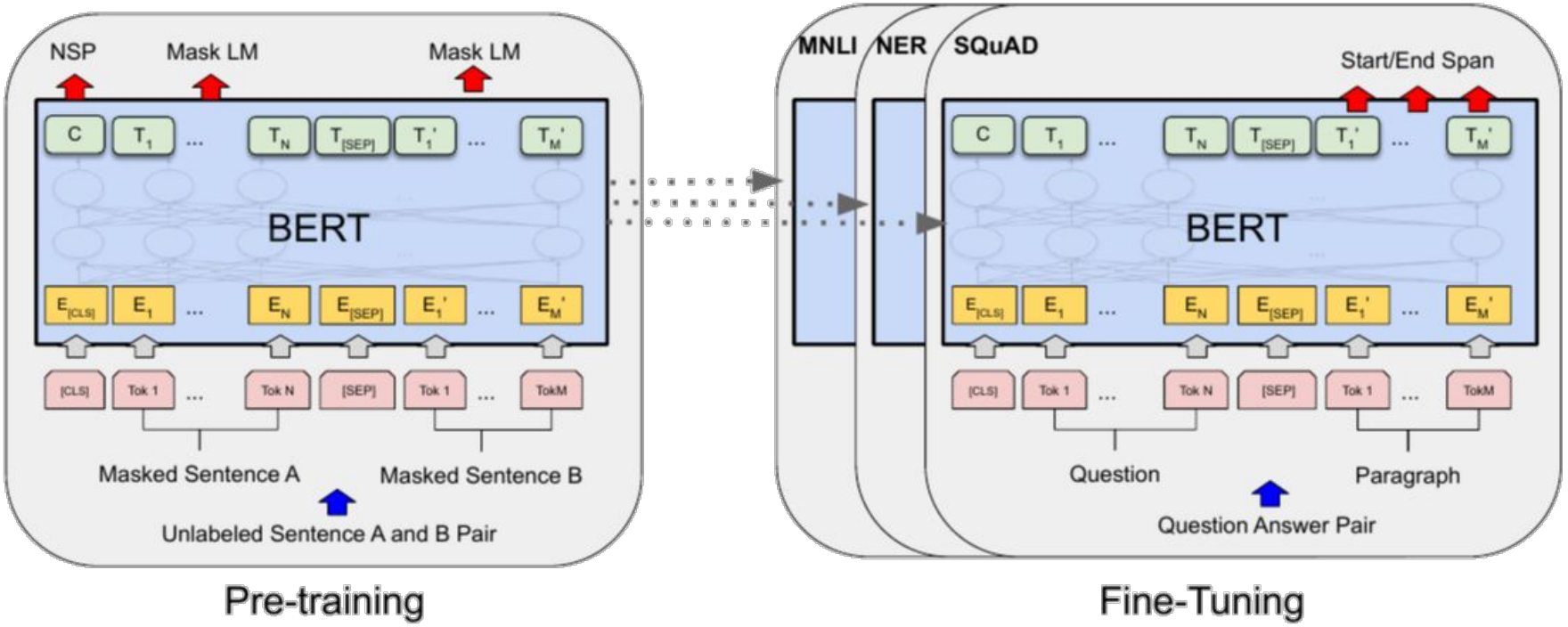
- În produsul $Q*K^T$, Q reprezintă unde acordăm atenție pe cheile K.

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

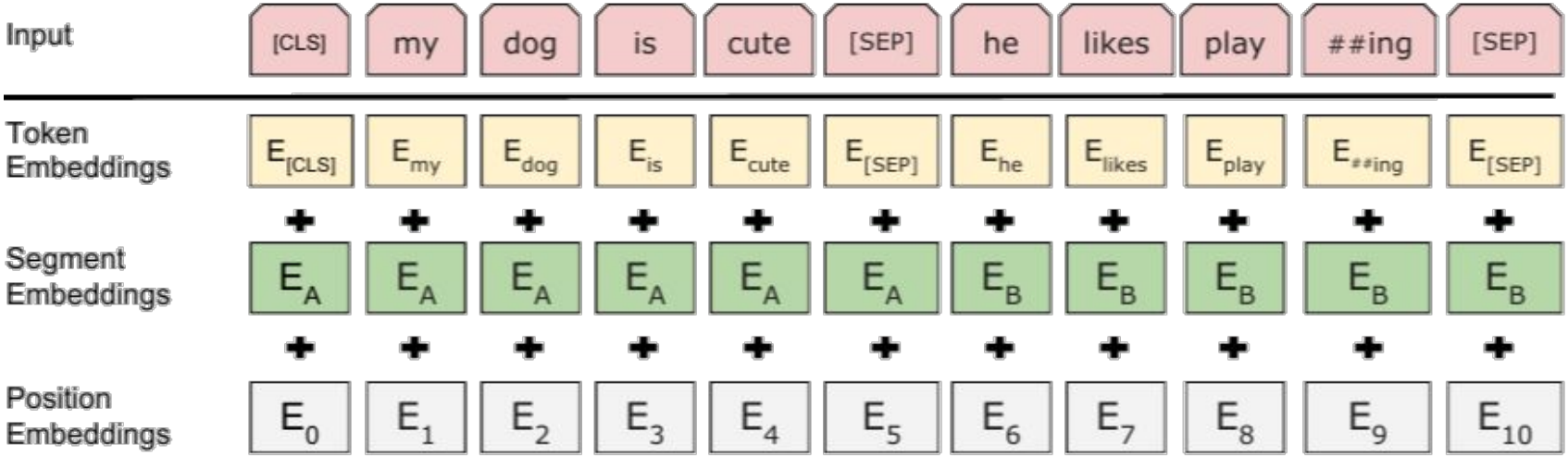
BERT - Transformer but bidirectional

- BERT folosește aceeași structură a transformer-ului, doar că adaugă câteva idei în plus:
  - Modifică intrarea adăugând 2 tokeni: [CLS] și [SEP], primul este folosit pentru noul task de NSP iar SEP separa 2 propoziții
  - Adaugă task-ul de NSP (Next Sentence Prediction)
  - Se folosește de MLM (masked LM) pentru a prezice tokenii în mod bidirecțional

BERT


NSP
Mask LM
Mask LM
C
T1
...
TN
T[SEP]
T1'
...
TM'
BERT
E[CLS]
E1
...
EN
E[SEP]
E1'
...
EM'
[CLS]
Tok 1
...
Tok N
[SEP]
Tok 1
...
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
Pre-training
MNLI
NER
SQuAD
Start/End Span
Question
Paragraph
Question Answer Pair
Fine-Tuning

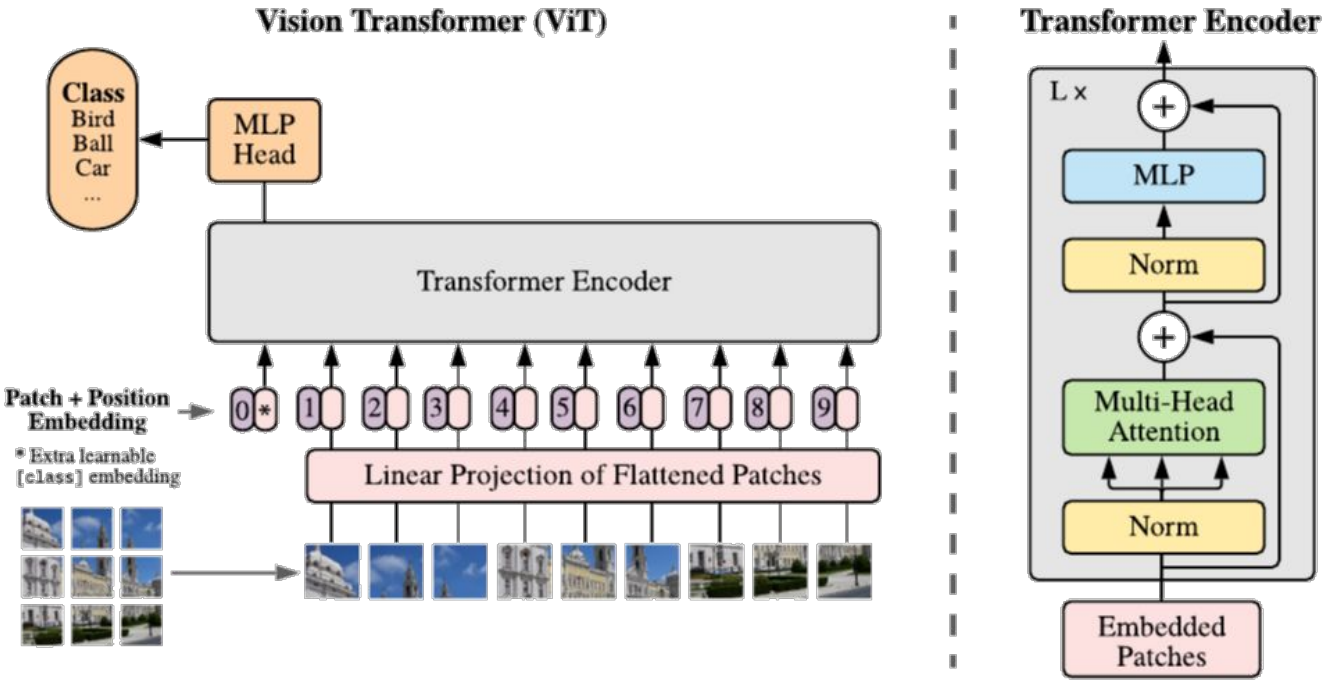
Intrare BERT



Vision Transformer(ViT)

- Este un transformer bazat pe BERT care a obținut unele rezultate state of the art, deci având o performantă mai bună decât CNN
- Folosește același token [CLS] ca la BERT pentru a prezice clasa imaginii, dar nu mai folosește [SEP]
- Pentru a primi ca input o imagine, aceasta este împărțită în patch-uri (e.g. modelul de pe huggingface primește imagini de 224x224 cu patch-size 16, adică un sequence 14*14 + 1 = 197)

Vision Transformer



Positional Encoding (ViT)

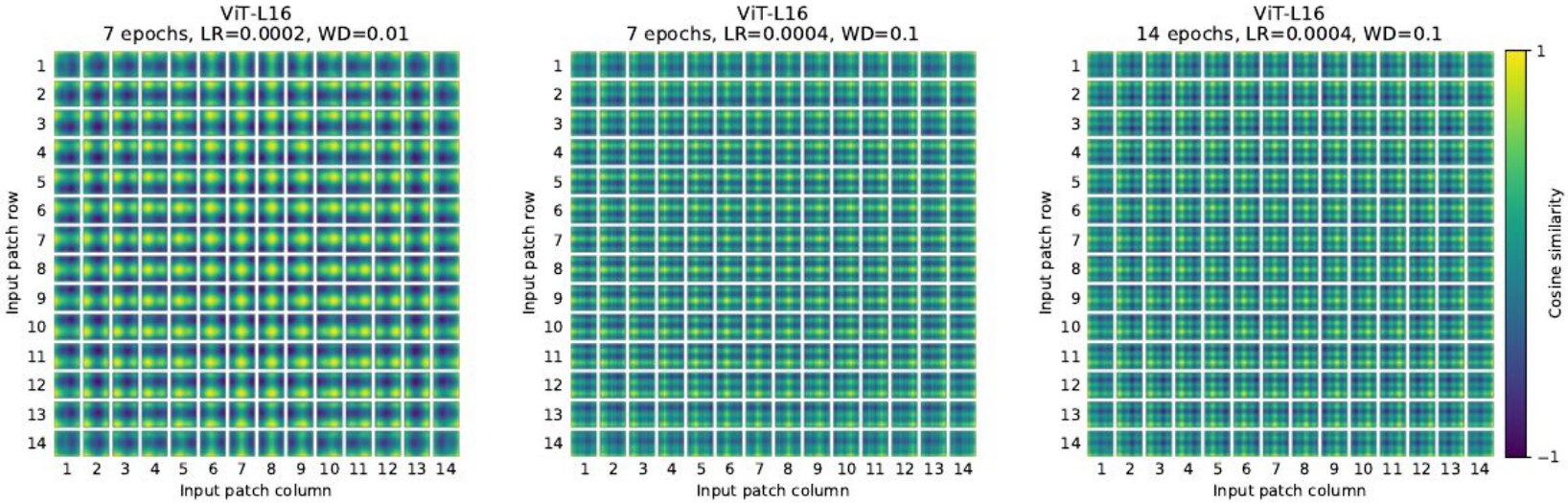


Figure 10: Position embeddings of models trained with different hyperparameters.

Pre-training ViT

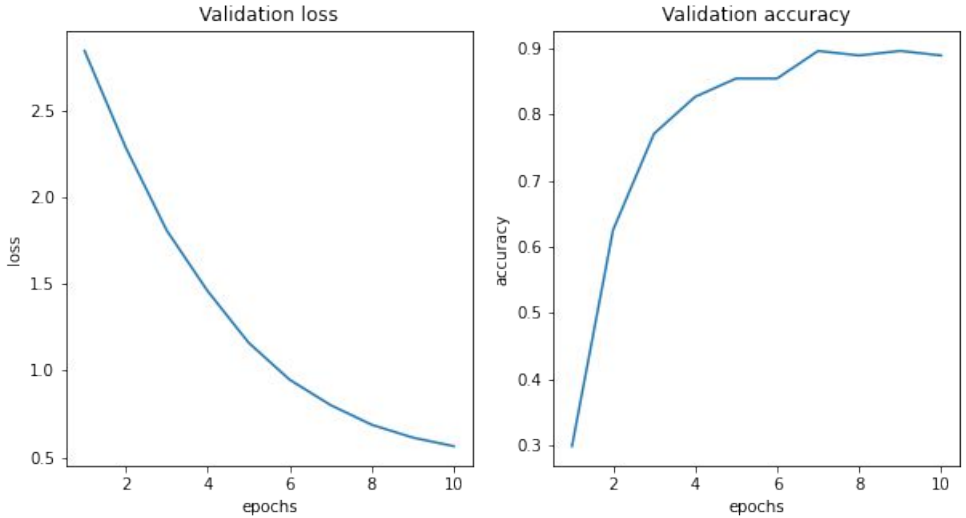
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

ViT pe clasificare si bounding box

- În proiectul acesta am folosit dataset-ul din laboratorul 5, American sign language letters, care implică clasificare și bounding box prediction
- Ideea principală este de a compara rezultatele obținute în laboratorul 5 cu cele obținute folosind un vision transformer
- Am luat modelul google/vit-base-patch16-224-in21k care e antrenat on imagenet-21k cu patch-uri de 16x16 si input image size 224

Model pre-antrenat cu clasificator

- Am încercat inițial doar clasificare folosind clasa VitForImageClassification care adaugă un clasificator la BERT
- Acest model obține 88.88% pe validare



Model cu clasificator custom

```python
class ClassificationBBoxTransformer(nn.Module):
    def __init__(self, backbone, num_classes):
        super(ClassificationBBoxTransformer, self).__init__()
        self.backbone = backbone
        self.classifier = nn.Sequential(
            nn.Linear(backbone.config.hidden_size, 512),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(512, num_classes)
        )
        self.bounding_box = nn.Sequential(
            nn.Linear(backbone.config.hidden_size, 512),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(512, 4),
            nn.Sigmoid()
        )

    def forward(self, x):
        features = self.backbone(x)
        features = features.last_hidden_state[:, 0, :]
        # print(features[:, 0, :].shape)
        return self.classifier(features), self.bounding_box(features)
```

- Considerând că trebuie făcute două task-uri trebuie să definim un model custom, care folosește ca backbone clasa ViTModel (ViT doar ca fără clasificator)

Fine-tuning ViT

- ViT este un model pre-antrenat deci trebuie să facem fine-tuning pe datele noastre pentru a putea prezice cu acuratețe
- ViT este un model care ocupă foarte mult VRAM, și fiind limitat hardware am decis să dau freeze la unele straturi (primele 6) combinând ideea de feature extraction cu fine-tuning

```
for param in vit_backbone.embeddings.parameters():
    param.requires_grad = False
for layer in vit_backbone.encoder.layer[:NUM_LAYERS_FROZEN]:
    for param in layer.parameters():
        param.requires_grad = False
```

Problemă loss-uri

- Avem 2 task-uri deci prin urmare avem și 2 loss-uri care de obicei tind să fie unbalanced:

```
Classification loss:
tensor(2.4068, device='cuda:0', grad_fn=<NllLossBackward0>)
Bounding box loss:
tensor(0.0210, device='cuda:0', grad_fn=<MseLossBackward0>)
```

- Deci loss-ul pentru clasificare este aproximativ 100x mai mare, deci vom introduce o constantă C care va echilibra loss-urile (împărțind classification loss la C)

Rezultate obținute cu resnet (lab5)

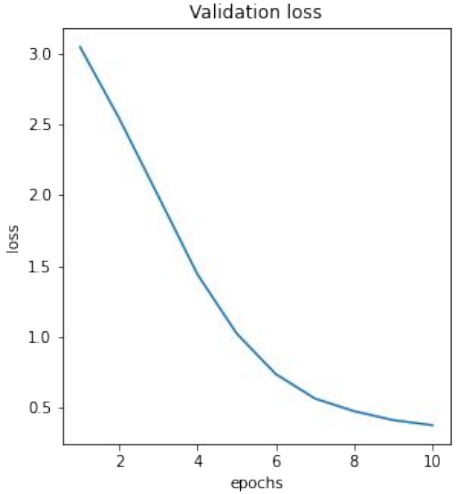
- Antrenare folosind resnet18 ca backbone pentru modelul definit cu un clasificator și un regressor

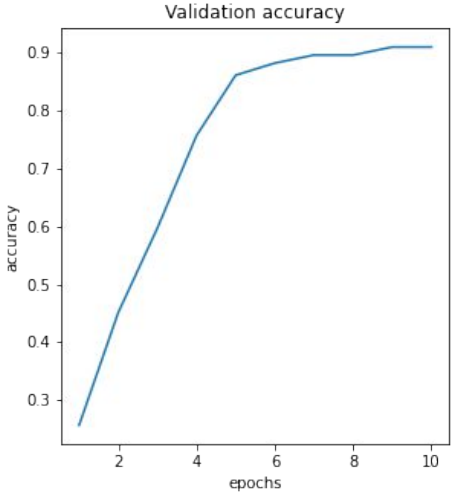
	Antrenare	Validare	IOU 50%	IOU 75%	IOU 90%
Acuratete	100%	**88.19%**	100%	84.72%	**13.19%**
Loss	0.0099	0.4073			

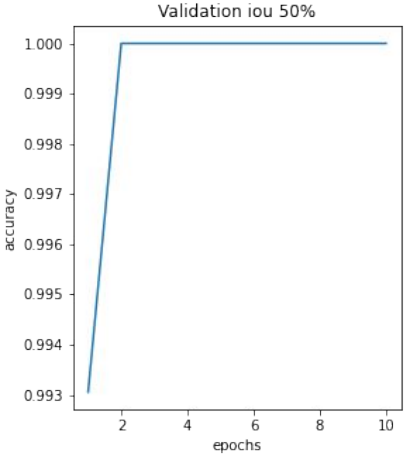
Rezultate ViT - fără C

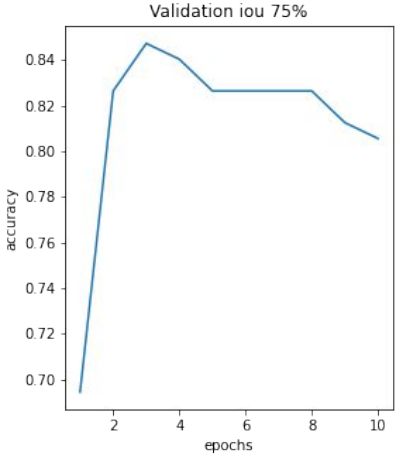
- Folosind transformer-ul descris mai sus am obținut rezultatele:
- Aici am considerat ViT fără variabila C și cu reducerea learning rate-ului (ReduceLROnPlateau) și gradient clipping (mici optimizări)

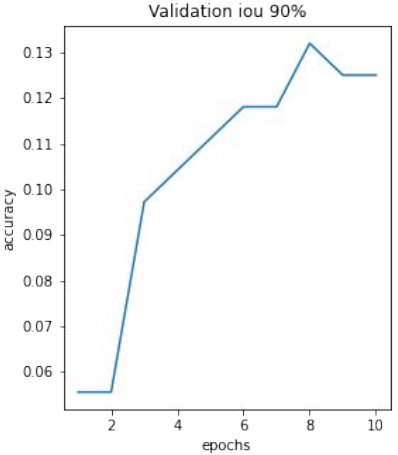
	Antrenare	Validare	IOU 50%	IOU 75%	IOU 90%
Acuratete	100%	**90.97%**	100%	80.55%	**12.5%**
Loss	0.0854	0.3724			


Validation loss
loss
epochs
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10


Validation accuracy
accuracy
epochs
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2
4
6
8
10


Validation iou 50%
accuracy
epochs
1.000
0.999
0.998
0.997
0.996
0.995
0.994
0.993
2
4
6
8
10


Validation iou 75%
accuracy
epochs
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0.70
2
4
6
8
10

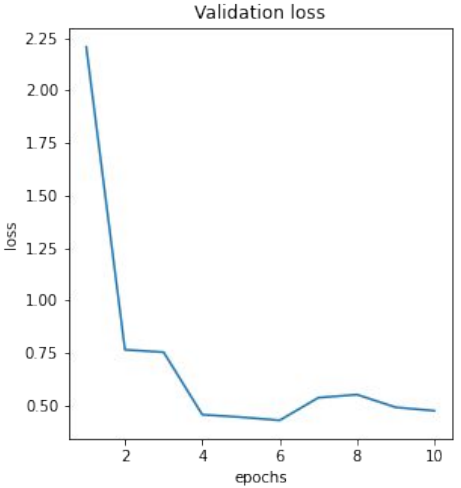

Validation iou 90%
accuracy
epochs
0.13
0.12
0.11
0.10
0.09
0.08
0.07
0.06
2
4
6
8
10

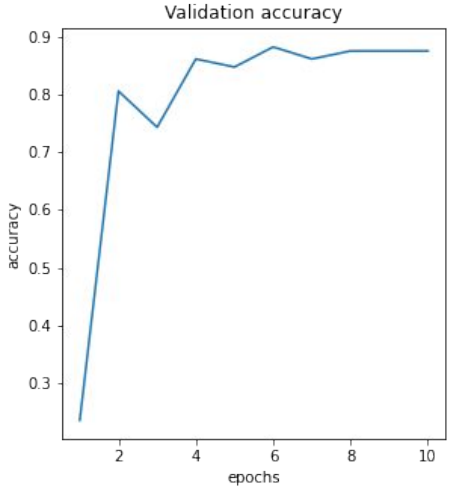
Rezultate ViT - cu C

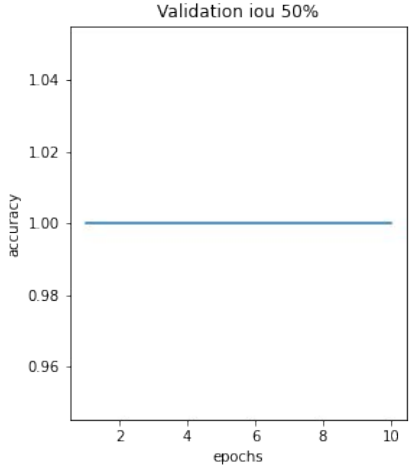
- Acum considerăm rezultatele obținute adăugând constanta C pentru echilibrarea loss-ului
- Se poate observa scăderea mică pe validare a acurateții clasificării dar crește foarte mult acuratețea pe bounding box, as expected

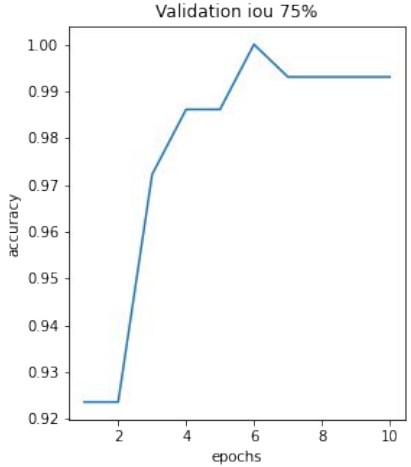
		Antrenare	Validare	IOU 50%	IOU 75%	IOU 90%
Cu Sigmoid	Acuratete	100%	**87.50%**	100%	99.3%	**63.88%**
	Loss	0.0006	0.4733			
Fara Sigmoid	Acuratete	100%	**88.19%**	100%	99.3%	**65.97%**
	Loss	0.0006	0.3549			

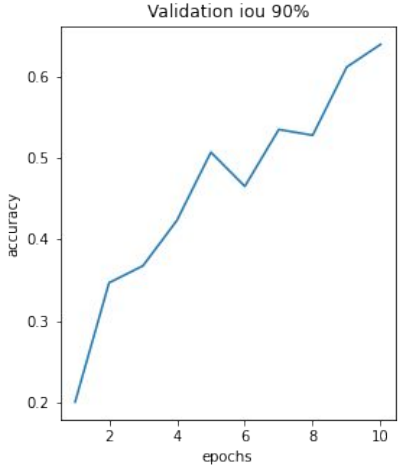
Cu Sigmoid:


Validation loss
loss
epochs

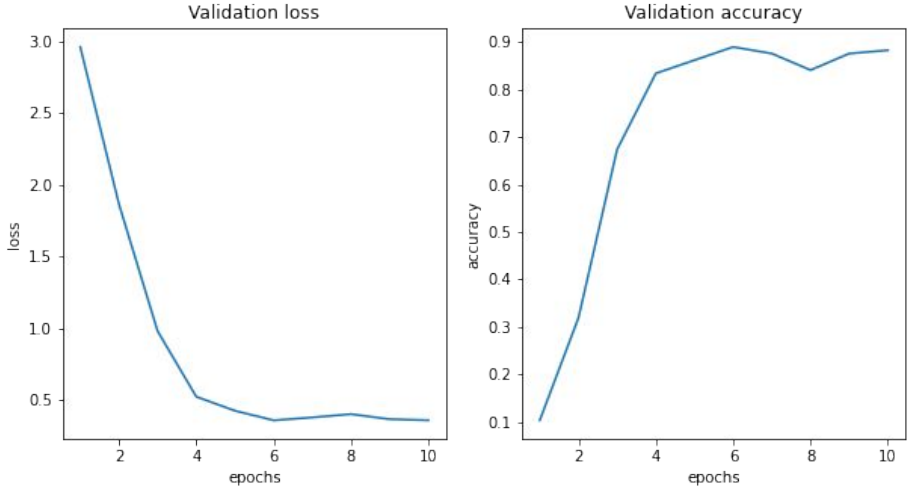

Validation accuracy
accuracy
epochs

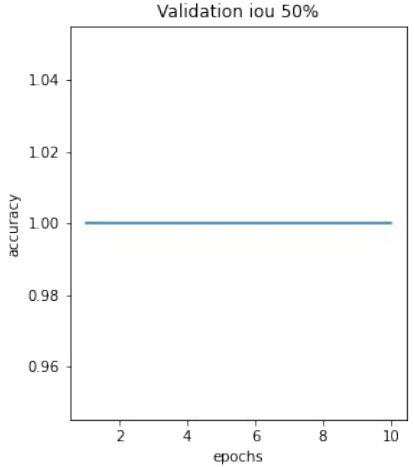

Validation iou 50%
accuracy
epochs

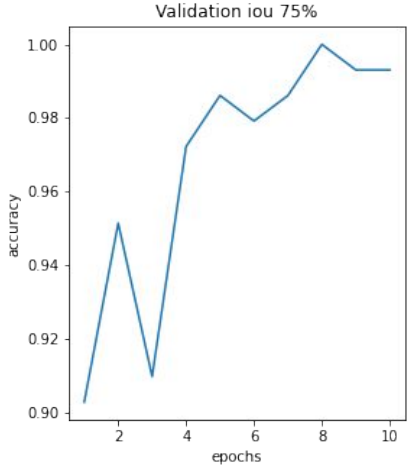

Validation iou 75%
accuracy
epochs

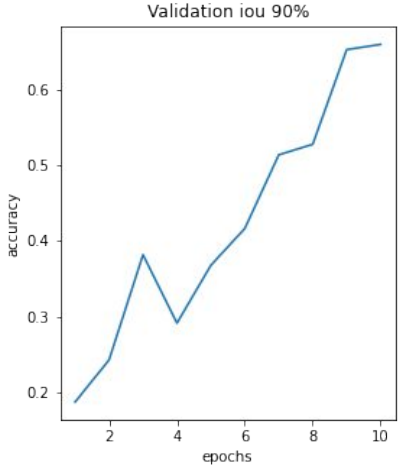

Validation iou 90%
accuracy
epochs

Fara Sigmoid:









Alte experimente și observații

- Rezultatele prezentate mai sunt sunt cele mai bune obținute, am încercat și alți parametrii precum: modificarea optimizatorului, diverse modele pentru clasificator și regressor, diferite learning rate-uri, număr diferit de layere înghețate
- Cel mai probabil se pot optimiza parametrii în așa fel încât acuratețea clasificatorului să fie 90%+ așa cum a fost înainte de adăugarea C-ului

Bibliografie

- https://arxiv.org/abs/1706.03762 (Vaswani et al. - Transformer)
- https://arxiv.org/abs/1810.04805 (Devlin et al. - BERT)
- https://arxiv.org/abs/2010.11929 (Dosovitskiy et al. - ViT)
- https://huggingface.co/docs/transformers/model_doc/vit

Vă mulțumesc!