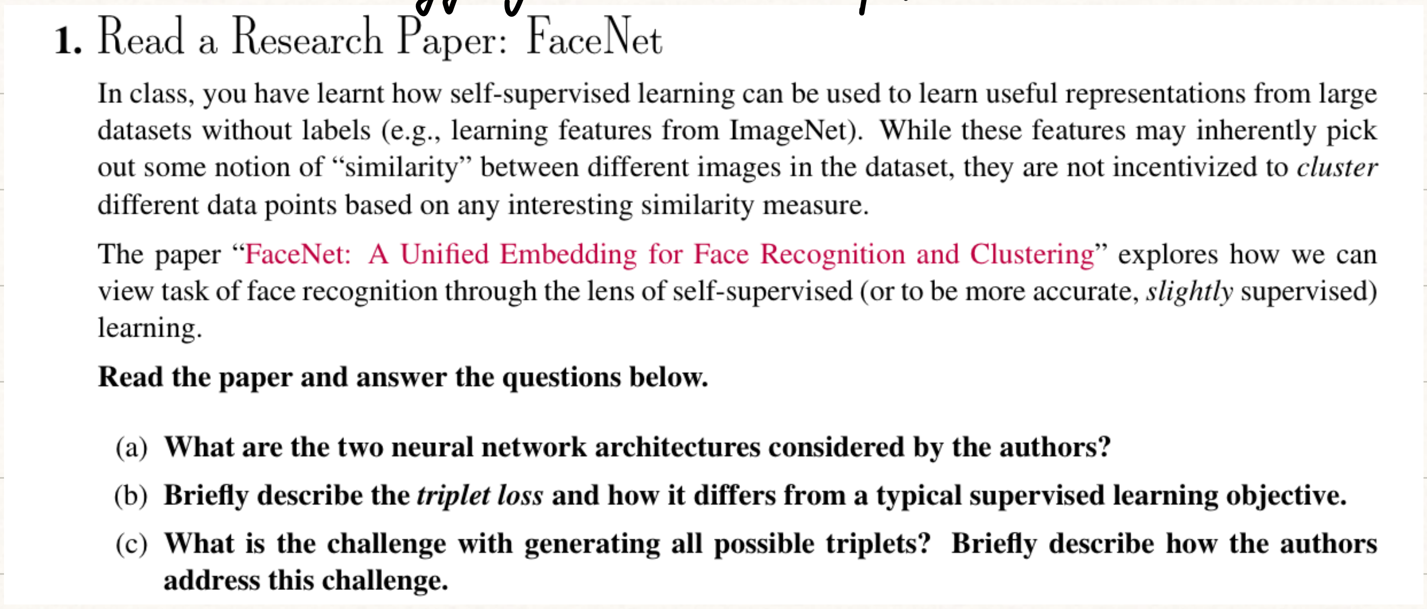
HW9.

Mengying Lin SID: 3038737132.



1. Read a Research Paper: FaceNet

In class, you have learnt how self-supervised learning can be used to learn useful representations from large datasets without labels (e.g., learning features from ImageNet). While these features may inherently pick out some notion of "similarity" between different images in the dataset, they are not incentivized to *cluster* different data points based on any interesting similarity measure.

The paper "FaceNet: A Unified Embedding for Face Recognition and Clustering" explores how we can view task of face recognition through the lens of self-supervised (or to be more accurate, *slightly* supervised) learning.

Read the paper and answer the questions below.

(a) **What are the two neural network architectures considered by the authors?**

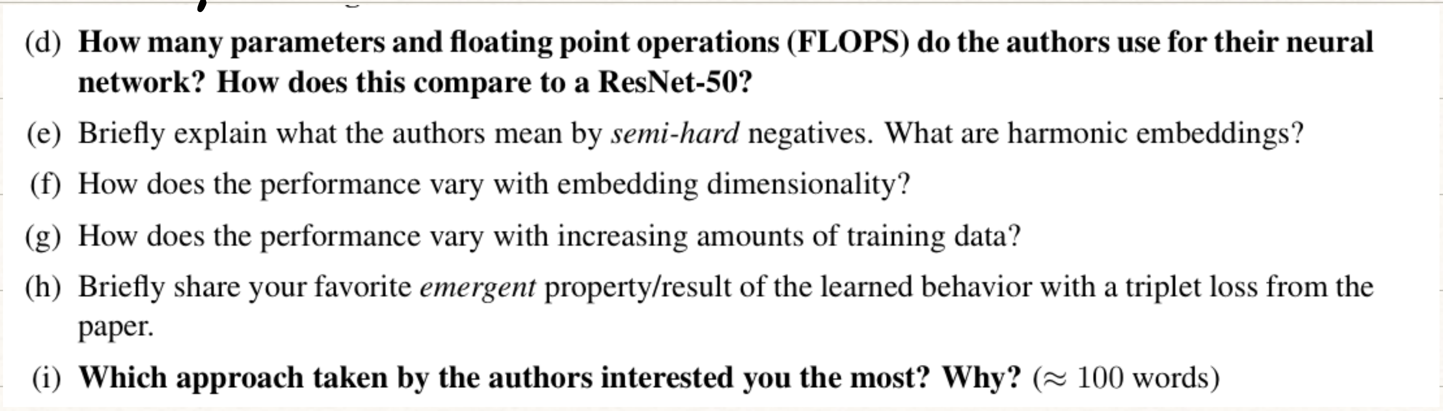
(b) **Briefly describe the *triplet loss* and how it differs from a typical supervised learning objective.**

(c) **What is the challenge with generating all possible triplets? Briefly describe how the authors address this challenge.**

a) Zeiler & Fergus model and Inception.

b) Triplet loss trys to enforce a margin between each pair of faces while traditional ones usually strive to encourages all faces of one identity to be projected onto a single pt in the embedding space.

c) This would result in many triplets that are easily satisfied.



(d) **How many parameters and floating point operations (FLOPS) do the authors use for their neural network? How does this compare to a ResNet-50?**

(e) Briefly explain what the authors mean by *semi-hard* negatives. What are harmonic embeddings?

(f) How does the performance vary with embedding dimensionality?

(g) How does the performance vary with increasing amounts of training data?

(h) Briefly share your favorite *emergent* property/result of the learned behavior with a triplet loss from the paper.

(i) **Which approach taken by the authors interested you the most? Why?** (≈ 100 words)

d). Zeiler & Fergus architecture:

140M parameters and 1.6B FLOPS.

Inception:

6.6M-7.5M parameters and 500M-1.6B FLOPS.

e). It removes the margin term α, which sets loose the restrictions making it easier to achieve.

Harmonic embeddings are generated by different models but are competible to each other.

f) Imcrease the dimensionality up to a certain degree will improve the performance. Futher increase sees no distinct improvement.

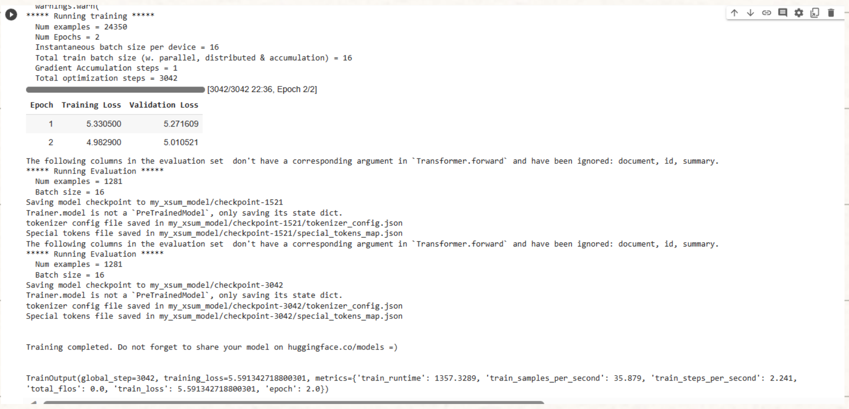
e) The more the training data is, the better the performance gets.

h) The learned face embeddings are invariant to variations in illumination, pose and facial expression.

i) Harmonic embedding. The embeddings generated by 2 different models are compatible and can be compared with each other.

3. Coding Question: Summarization (Part I)

(a) **Please report your final validation accuracy after 2 epochs, along with screenshots of the training loss and the validation loss displayed on Tensorboard.**



```
  warnings.warn(
***** Running training *****
  Num examples = 24350
  Num Epochs = 2
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 3042
```

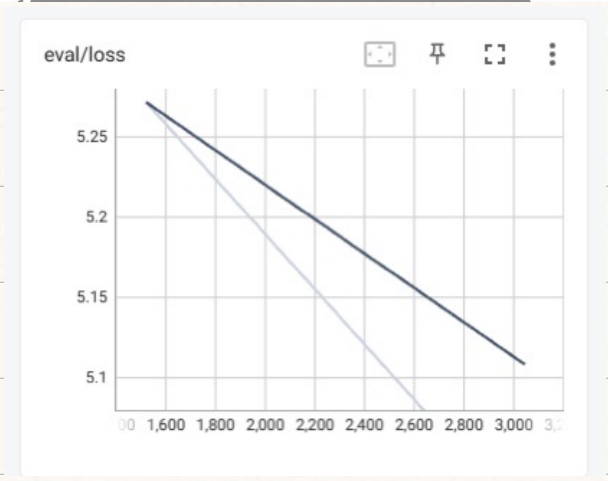
[3042/3042 22:36, Epoch 2/2]

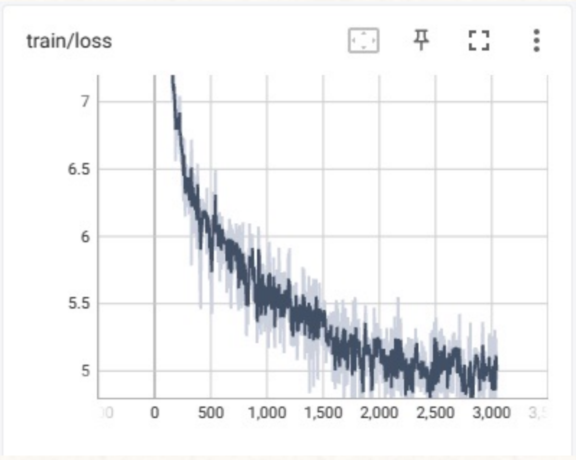
Epoch	Training Loss	Validation Loss
1	5.330500	5.271609
2	4.982900	5.010521

```
The following columns in the evaluation set  don't have a corresponding argument in `Transformer.forward` and have been ignored: document, id, summary.
***** Running Evaluation *****
  Num examples = 1281
  Batch size = 16
Saving model checkpoint to my_xsum_model/checkpoint-1521
Trainer.model is not a `PreTrainedModel`, only saving its state dict.
tokenizer config file saved in my_xsum_model/checkpoint-1521/tokenizer_config.json
Special tokens file saved in my_xsum_model/checkpoint-1521/special_tokens_map.json
The following columns in the evaluation set  don't have a corresponding argument in `Transformer.forward` and have been ignored: document, id, summary.
***** Running Evaluation *****
  Num examples = 1281
  Batch size = 16
Saving model checkpoint to my_xsum_model/checkpoint-3042
Trainer.model is not a `PreTrainedModel`, only saving its state dict.
tokenizer config file saved in my_xsum_model/checkpoint-3042/tokenizer_config.json
Special tokens file saved in my_xsum_model/checkpoint-3042/special_tokens_map.json


Training completed. Do not forget to share your model on huggingface.co/models =)


TrainOutput(global_step=3042, training_loss=5.591342718800301, metrics={'train_runtime': 1357.3289, 'train_samples_per_second': 35.879, 'train_steps_per_second': 2.241,
'total_flos': 0.0, 'train_loss': 5.591342718800301, 'epoch': 2.0})
```





4. Coding Question: Visualizing Attention

Please run the cells in the Visualizing_BERT.ipynb notebook, then answer the questions below.

(a) Attention in GPT: Run part a of the notebook and generate the corresponding visualizations

i. **What similarities and differences do you notice in the visualizations between the examples in this part?** Explore the queries, keys, and values to identify any interesting patterns associated with the attention mechanism.

ii. **How does attention differ between the different layers of the GPT model? Do you notice that the tokens are attending to different tokens as we go through the layers of the network?**

i) Similarities: In attention machenism, only the similarities between one word and the words ahead of it will be taken into consideration. Namely, the process is causal.

Differences :

ii) The keys and queries are constantly changing.

(b) BERT pays attention: Run part b of the notebook and generate the corresponding visualizations.

i. Look at different layers of the BERT model in the visualizations of part (b) and identify different patterns associated with the attention mechanism. Explore the queries, keys, and values to further inform your answer. **For instance, do you notice that any particular type of tokens are attended to at a given timestep?**

ii. **Do you spot any differences between how attention works in GPT vs. BERT? Think about how the model architectures are different.**

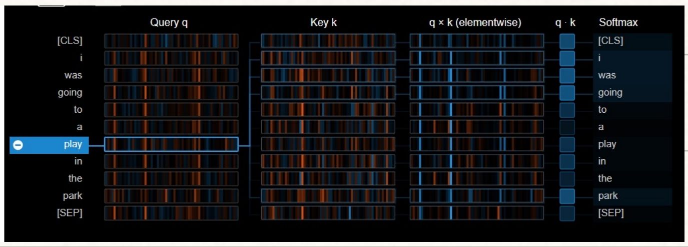
iii. For the example with syntactically similar but definitionally different sentences, look through the different layers of the two BERT networks associated with sentence a and sentence b, and take a look at the queries, keys, and values associated with the different tokens. **Do you notice any differences in the embeddings learned for the two sentences that are essentially identical in structure but different in meaning?**

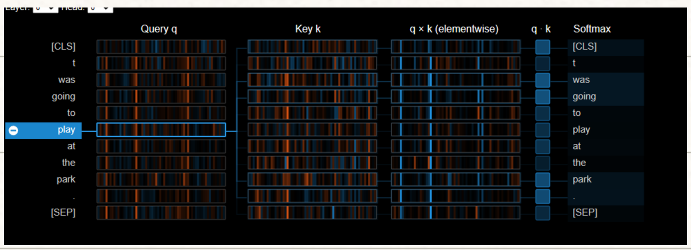
iv. **For the pre-training related examples, do you notice BERT's bi-directionality in play? Do you think pre-training the BERT helped it learn better representations?**

i In first few layers. attention is more focused on local patterns. In the later layers, it is more global.

ii. The attention in GPT is autoregressive. and the decoder can only attend to previously generated tokens, while for BERT. each token will attend to every other tokens. This is because GPT only uses decoder part while BERT uses the encoder.

iii. Yes. e.g. In layer 0, attention head 0, the k.q.v went like this:





The correlation between "play" and other words are not the same in the 2 cases.

iv. Yes. Every token will be paired with any other tokens. If in layer i token i attends to token j. we'll find token j attends to i as well.

(c) BERT has multiple heads!: Run part c of the notebook and generate the corresponding visualizations.

i. **Do you notice different features being learned throughout the different attention heads of BERT? Why do you think this might be?**

ii. **Can you identify any of the different features that the different attention heads are focusing on?**

i. The features learned in one sentence are stronger and those learned across two sentences are weaker and more concentrated. (i.e. Usually the attention between one particular word in sentence A ✓

with others in sentence B are more distinctive than other words

in sentence A do.

(d) Visualizing untrained attention weights

i. **What differences do you notice in the attention patterns between the randomly initialized and trained BERT models?**

ii. **What are some words or tokens that you would expect strong attention between? What might you guess about the gradients of this attention head for those words?**

i. For untrained BERT, the attention across each pairs

are more even and there are less distinctive patterns.

ii. I think "sad" and "happy" are likely to gain stronger

attention and the gradients of corresponding attention head

will be greater.

(e) **Were you able to identify interesting patterns in the visualizations?** If yes, please share some examples (describe in text or paste a screenshot). If not, feel free to use this space for your frustrations.

① The attention in Transformer is strictly causal while

that in BERT connects every pair of tokens.

②. Pretrained models can "pay attention" in a more targeted

manner.