EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Discussion 11

1. Finetuning Pretrained NLP Models

In this problem, we will compare finetuning strategies for three popular architectures for NLP.

(a) **BERT** - encoder-only model

(b) **T5** - encoder-decoder model

(c) **GPT** - decoder-only model

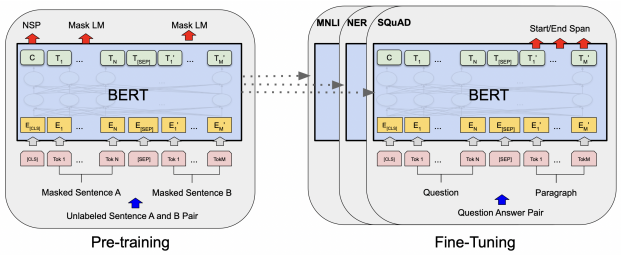

Figure 1: Overall pre-training and fine-tuning procedures for BERT.

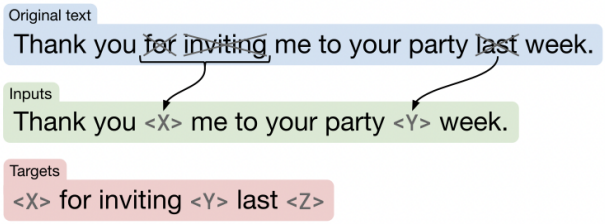

Figure 2: T5 Training procedure

(a) For each of the three models, state the objective used for pretraining.

Solution: BERT uses two training objectives: a masked token prediction objective (i.e. a cross-entropy loss for predicting input tokens which were masked out), and a next sentence prediction objective which predicts whether the two sentences passed in are sequential or not. This is shown in Figure 1 (Followup works such as RoBERTa only use the masked prediction objective.)

T5 uses a slightly different masking objective. Spans of multiple words are masked out from the encoder. The decoder must predict the tokens for each span, as shown in Figure 2.

GPT is trained using using a next-token classification loss.

(b) Consider the MNLI (Multi-Genre Natural Language Inference Corpus) task. It provides a passage and a hypothesis, and you must state whether the hypothesis is an entailment, contradition, or neutral.

EXAMPLE:
Passage: At the other end of Pennsylvania Avenue, people began to line up for a White House tour.
Hypothesis: People formed a line at the end of Pennsylvania Avenue.
Classification: entailment

(i) With each of the 3 models, state whether it is possible to use the model for this task with no finetuning or additional parameters. If so, state how.

Solution: With GPT and T5, you could accomplish this by **few-shot prompting**. Input a set of examples (passage, hypothesis, and answer), followed by the passage and hypothesis for the current question. The decoder would then predict the answer.
It is technically possible to use BERT for this task by inputting a prompt, masking a token for the answer, and predicting it, but BERT is not good at learning through prompting.[1].

(ii) With each of the 3 models, state how you would use the model for this task if you were able to add additional parameters and/or finetune existing parameters.

Solution: With GPT and T5, you could input the passage and hypothesis, then finetune the model to predict the answer as the next token. Note that with large models it is often common to only finetune a subset of the model parameters.[2]
With BERT, you input a sequence to the model: [<CLS>, passage, <SEP>, hypothesis]. Remove the output layer of the network and replace it with a classification head on the CLS output token. (Some approaches also make use of the other tokens, such as by taking a mean across the other output tokens and concatenating them with the CLS token.) You can either finetune all parameters or only the parameters of the classification head.
(Students may suggest other answers which are valid too - e.g. you could extract features from GPT by concatenating token features from the last couple of layers and training a classification head on top of them, but these may not perform as well).

(c) Compare and contrast the ways we use pretrained representations in BERT to the way we use pretrained autoencoder representations.

Solution: When you use a single layer of BERT as your representation, the bottom of the network serves the role of the AE encoder (the piece you use to encode inputs for the downstream task), and the final layer(s) serve the role of the AE decoder (the piece which is used in training then thrown away). However, NLP applications sometimes use features from multiple layers in the model, whereas AEs typically use a single layer representation.

2. Vision Transformer

Vision transformers (ViTs) apply transformers to image data by following the following procedure:

(a) **Split image into patches** - The original ViT paper split images into a 16x16 grid of patches.

(b) **Convert each patch into a single vector** - In the original paper, they flattened the patch and applied a linear projection.

(c) **Stack the patches into a sequence, concatenate a CLS token, and add in positional embeddings.** Absolute learned positional embeddings are most common here.

[1] For new work modifying BERT to benefit from prompting, see https://arxiv.org/pdf/2201.04337.pdf
[2] For a comparison between of several parameter-efficient finetuning methods, check out the experiments in this paper: https://arxiv.org/pdf/2205.05638.pdf

(d) **Pass the sequence through a transformer as usual.**

Below is a diagram of ViT for supervised image classification.

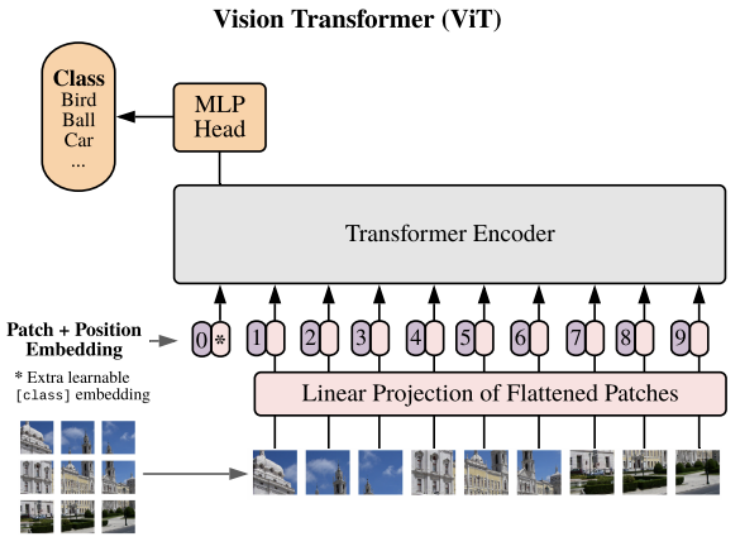

Figure 3: Vision Transformer (Link to Google blog)

(a) Does it matter which order you flatten the sequence of patches?

Solution: No, the positional encoding will tell the model where the patch came from, as long as we use the same flattening order (ie. row-order or column-order) throughout training.

(b) What is the complexity of the vision transformer attention operation? Assume you have an image of size HxW and patches of size PxP. Only consider the time of the attention operation, not the time to produce queries, keys, and values. Queries, keys, and values are each size D.

Solution: We have (H/P)*(W/P) patches, so the complexity is $O(seq^2 D) = O((HW/P^2)^2 D$.

(c) What is the receptive field of one sequence item after the first layer of the transformer? How does this compare to a conv net, and what are the pros and cons of this?

Solution: After a single transformer layer, the receptive field is the entire image. In contrast, a conv net's receptive field grows slowly through the layers. This change allows transformers to pick up on spatially distant patterns more easily. However, the conv net's inductive bias toward spatially local patterns can be helpful for learning, especially when training from scratch on a small dataset. Some transformers have also explored only attending to spatially-local patches in early layers of the transformer to recover this inductive bias.

(d) If we forgot to include positional encodings, could the model learn anything at all? State one task where a model could perform well without positional encoding, and one task where it would do poorly.
Solution: The model would see the image as a bag of patches (similar to a bag of words in NLP). It could still perform well on tasks where patches are sufficient to understand the content of the image (e.g. classifying trafic lights as red, yellow, or green), but it would perform poorly on patches that rely on the order of patches (e.g. reading text from a screen).

(e) If you wanted to add a few conv layers into this achitecture, how would you incorporate them?

Solution: There's no one unique solution to this. One way is to first apply a few conv layers to the input image, and then split the resulting feature map into patches for the rest of the ViT operations (Xiao et al). Research found that applying convolution for early "patchify" layer leads to more robust ViT training. Another way is to apply a small convolution filter after each ViT block to "reduce the spatial dimension" of the feature map (Xie et al), which can be thought of as intentionally adding a *locality bias* regularly after every few self-attention layers, which have low inductive bias and a receptive field of global.

(f) How would you use this architecture to do GPT-style autoregressive generation of images?

Solution: You would need to pass the transformer's output into a decoder which can generate image patches. Like with language, you could then feed the generated patch back into the model to generate the next patch. Train with MSE loss against the ground-truth patches. (Or cross-entropy loss if you're using discrete tokens, e.g. Parti `https://parti.research.google/`. I don't think discrete tokenization e.g. VQVAE has been covered yet.)

(g) So far the vision transformer training procedure has been fully-supervised. Yet we know that most of the available data are unlabelled. How can we do BERT-style self-supervised **representation** learning with vision transformers?

- *Hint 1*: Think about how BERT is trained. How should the input image be modified? What should be the target?)
- *Hint 2*: ViT in this question only has an encoder. For BERT-style training, you will need a decoder. Why is this the case?

Solution:

This question also does not have a unique solution. Instead, the following solution provides an overview of proven effective methods in past research.

BERT is able to obtain useful representations for NLP tasks using two pretext tasks: (1) mask prediction (2) next sentence prediction, with the mask prediction objective being the more effective one. In vision transformer, the analog of mask prediction will be **predicting the masked patches**.

Specifically, we can mask out, say, 50% of the input patches and treat the masked image as input to our model. The model will be trained to predict the masked region of the original image, by optimizing a reconstruction loss (e.g. mean squared error).

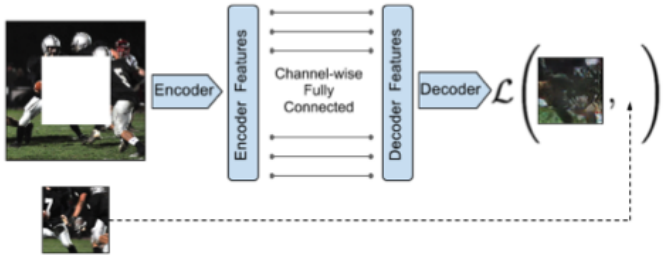

Figure 4: Diagram of Context Encoder

This objective of predicting masked patches or pixels has a techincal name called **inpainting**, and the aforementioned procedure is a simplified version of **Context Encoder** (Pathak et al), which takes in a masked image (with masked pixels filled with mean value of the image) and predicts the missing

region. While Context Encoder was an effective self-supervised representation learning method at the time, there's a design choice that adversely impact its performance: **mask tokens should not be fed into the image encoder**.

The intuition behind why mask tokens should not be fed into image encoder is that at test-time, real images do not contain such a huge portion of mask values. This creates a huge distribution difference between train- and test-time. To learn useful representations, a better design would be to only feed the encoder *real* patches and then append the mask tokens / values right before the decoder input for image reconstruction.

This is the idea of Mask Autoencoder (He et al) or MAE. MAE uses a Vision Transformer encoder-decoder structure, which takes in an image, patchify the image, randomly mask out 75% of the patches, pass the un-masked patches into the encoder, and lastly append the mask tokens right before the decoder input.

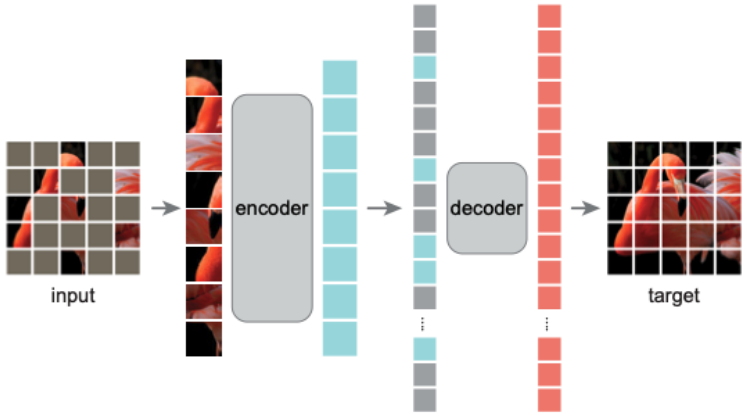

Figure 5: Diagram of Masked Autoencoder

Contributors:

- Olivia Watkins.
- Anant Sahai.
- Kevin Li.
- CS 182/282A Staff from previous semesters.

