EECS 182 Deep Neural Networks
Fall 2022 Anant Sahai

Final Review: Transformers, Finetuning, and Prompting

1. Transformers

(a) Let's say you are training a navigation robot using a robotic navigation task. The user describes in language an object in the house, and the robot must navigate the house to find it. The robot sees an image from its front-facing camera at each timestep, and it outputs one of three movement actions: [LEFT, RIGHT, FORWARD]. You have a dataset of example robot trajectories. Describe how you would use a transformer for this task. (What architecture? What is inputted to the transformer? What is outputted?)

Solution: Answers may vary, but here is one reasonable setup:

- Use an encoder/decoder model (e.g. T5).
- The target object gets encoded in the encoder. We can use a pretrained T5 language model and encode the inputs using the T5 tokenizer.
- The history of the trajectory gets passed into the decoder. We can encode input images using a small ViT or conv net (or just by linearly projecting patches) and encode past actions using learned token embeddings.
- The decoder uses autoregressive masking and predicts the aciton tokens (so each action is predicted based on the previous images, actions, and the target object).
- The final layer outputs three logits (one for each action) and is trained using the cross-entropy loss on the training dataset.

Robotics application, how use transformer for this?

(b) Why are subword tokenizers (e.g. WordPiece encodings or byte pair encodings) preferred over word tokenizers? **Solution:** Word tokenizers will map all words not in the vocabulary to <UNK>. This would make it hard to handle words containg misspellings, unusual names, or sTRaNgE f_o_r_m_a_t_t_i_n_g. Also, if <UNK> tokens are common in the dataset, the model may output a high proportion of <UNK>s.

2. Finetuning

(a) If you pretrain using a masked autoencoder, when you finetune the autoencoder encoder for downstream tasks, do you still mask the inputs? **Solution:** Typically you don't do this, though i's possible as a form of data augmentation in the downstream task.

(b) Let's say you want to train an LSTM encoder/decoder model on translating from English to Spanish using a paired English/Spanish training set. You also have a much larger corpus of unpaired English sentences. Describe one way to pretrain the LSTM encoder using the unpaired data. **Solution:** One option is to train an LSTM encoder/decoder model which acts as an autoencoder. An English sentence is fed into the encoder (possibly with noise or masking), and the decoder resconstructs the original sentence. Hopefully this will train the encoder to learn good representations of English sentences, so this encoder can be finetuned on the paired data.

(c) For each of the following finetuning problems, describe whether you should prefer to use (a) feature extraction (also called linear probing), (b) full finetuning, (c) hard prompting, or (d) soft prompting.

(a) You are using a 175B parameter language model for a question-answering task. You have a dataset of 100k examples. **Solution:** Soft prompting is preferred here, since when you have lots of data points (like here), soft prompting outperforms hard prompting.

(b) You are using a 90B parameter language model for a spam classification task. You have a task description but no training data. **Solution:** If you have no training data, hard-promting is the only valid option.

(c) You are using a 1B parameter conv net pretrained on ImageNet for wildlife classification task. You have 100 training examples. **Solution:** Feature extraction is probably the best. We can't (yet) use prompting with conv nets. With so few training examples, full-finetuning isn't a great option either since finetuning may distort the pretrained features and result in overfitting.

(d) You are using a 1B parameter vision transformer pretrained on ImageNet for an X-Ray fracture localization task. You have 100M training examples. **Solution:** You should use full finetuning. Since the downstream task is fairly different from the pretraining task, and the dataset is pretty large, full finetuning is likely to result in better performance.

3. Prompting

(a) Typically, when you create a soft prompt for use with a GPT model, you prepend the prompt to the left of the input. Could you also get good performance by appending it to the right? **Solution:** This could still work. Prompts are often appended at the beginning so that the input tokens can attend to the prompt, but if the prompt is on the right, the sequence items corresponding to the prompt can still attend to the other input tokens.

(b) Let's say you would like to use hard prompting with a large GPT model. You have a dataset of 10 thousand training examples for your downstream task. Would it be a good idea to include all of these examples (except for a held-out validation set) in your prompt? **Solution:** Probably not, for a few reasons:

- The dataset is likely longer than the sequences the transformer was trained on. If you are using learned positional encodings, you cannot run the model on longer sequences than were seen before. If you use fixed (e.g. sinusoidal) positional encodings, you can use longer sequences, but performance might degrade if the sequence length is very out of distribution from what is seen before.
- Past some point, performance is unlikely to keep getting better as you add more examples.
- The longer your prompt is, the slower and more computationally expensive it is to run the model. You may also eventually run into memory errors.

