Default Project 1 - DD2424, 2025

Teachers and TAs of DD2424 2025 and earlier

1 Explore Transfer Learning

The first default project will explore the concept of transfer learning. This is one of the most common use cases of deep learning - download a pre-trained model and then adapt it to your dataset. Please visit the tutorial FINETUNING TORCHVISION MODELS, this gives an overview of how to perform fine-tuning of with a pre-trained ConvNet within PyTorch. Another tutorial is given at FastAI Computer Vision tutorial.

1.1 Basic project to get E

- Download a pre-trained modern ConvNet such as: ResNet18, ResNet34,...
- Download the dataset The Oxford-IIIT Pet Dataset
- Replace the final layer of the pre-trained ConvNet to solve the binary classification problem of recognising pictures of Dog Vs Cat. Fine-tune the replaced final layer with the Pet Dataset's training data. Use Adam or NAG optimizer. Without too much effort you should be able to get very high performance ($\geq 99\%$ test accuracy) on this binary classification task.

  Note you will have to check what spatial size of input your pre-trained network can cope with. The default ResNet architectures have a global average pooling layer, just before the final fully connected output layer, that produces a feature vector of fixed size independent of the spatial extent of the input image. Thus you just have to ensure that the amount of down-sampling implemented during the ResNet does not make the spatial extent of the feature maps disappear for the later layers given the size of your input images. If your images are too small and this will happen then you should re-size them to the smallest acceptable size so that your computational effort is minimized. The normal procedure is to resize the image with one scale factor, to maintain the image's aspect ratio, so its shortest side is re-scaled to the target length.

- Next your goal is to solve the multi-class classification problem of recognising the breed of cat or dog. In this case you have to replace the final layer to have 37 outputs. As this multi-class problem is harder than the binary classification problem you have just solved, you will have to do more work and you should fine-tune more of the network as opposed to just the replaced final layer. You should explore each of the following issues when fine-tuning and use performance on a validation set in tandem with your computational budget to decide when to end training.

  - **Strategy 1; Fine-tune l layers simultaneously** Fine-tune the last l layers of the network (+ the classification layer) from the start of training. For the first experiment set $l = 1$, the next one re-fine-tune the pre-trained network but with $l = 2$, then $l = 3$

  until $l = L$ where L is defined by your available compute and also seeing when adding more layers results in only minimal or no changes.
  - **Strategy 2: Gradual un-freezing** Gradual unfreeze the layers of the network during fine-tuning. This strategy involves fine-tuning the network in stages. Start with the last few layers and then progressively unfreeze earlier ones. Is there any significant difference in the final performance or training time for these two strategies?
  - Different learning rates and/or learning rate schedulers for different layers.
  - Benefit of applying data augmentation during training (flip, small rotations, crops, small size scaling) and also L2 regularization.
  - Effect of fine-tuning or not the batch-norm parameters and updating the estimate of the batch mean and standard deviations on the final performance on the new dataset.

  After these experiments you should be able to get a final test accuracy of $\sim 95\%$ (note this number is a guideline and not a strict). You do not have to use the Pet Dataset. You are free to use another dataset but it should have comparable or greater difficulty than the Pet Dataset w.r.t. number of classes and size of images.

- **Fine-tuning with imbalanced classes** For the last exercise check what happens if you have imbalanced classes and try to fine-tune. One option would be to just use 20% of the training images for each cat breed. If you train with the normal cross-entropy loss, what happens to the final test performance on the classes with limited data? You should then try a strategy such as weighted cross-entropy and/or over-sampling of the minority classes to compensate for the imbalanced training set.

1.2 Extending the basic project to get a higher grade

Once you have explored relatively thoroughly the basic project then your group can add extensions to aim for a higher grade.

1.2.1 From E $\rightarrow$ D/C

If you are aiming for a D or C then here are some extensions you could apply to investigate if it was possible to improve performance.

- Explore using deeper networks than you used in the basic project. Does the deeper network help? Is it trickier to fine-tune? Do you even need to fine-tune the earlier layers? Do you need to change the optimizer to for instance AdamW to ensure good training and perform more L2 regularization?

- Explore the idea of **catastrophic forgetting**. It has been observed that if you fine-tune a pre-trained network to a dataset with different characteristics to the pre-training dataset then the network will progressively adapt its feature representations to the new dataset and possibly not maintain those learnt from the original dataset. When this happens it is known as **catastrophic forgetting**. You could consider your cat Vs dog classifier as your pre-trained network. A dataset potential distinct to ImageNet and the Pets dataset is the 102 Category Flower Dataset. You should try and see if you can induce catastrophic forgetting by “agressively” fine-tuning your cat Vs dog network to this new dataset. You may need a long training run. Ensure you get good performance on the new dataset and then check if you then re-train the classification layer for the Dog Vs Cat problem on the newly fine-tuned network if you can still get as good performance as you had before the fine-tuning to the flower dataset.

- Just fine-tune the batch norm mean and standard deviation and keep the weights of the layers (except the final layer) frozen and see if you can improve results.
- Add more sophisticated data-augmentations such as random erasing, CutMix or MixUp to help with regularization.

Assuming the write-up and presentation of the material is good, how many extensions are needed to achieve the desired grade? This is a tricky question as exploring one extension in depth can be considered equivalent to exploring many extensions superficially. So for example exploring CutMix and its benefits thoroughly should be more than sufficient to get a C, while a cursory investigation of the benefits of a deeper network and more simple regularization may just about make the cut for a D grade.

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal.

1.2.2 From E $\rightarrow$ B/A

If you are aiming for a B or A then here are some possible avenues to explore to extend the basic E project (You do **not need** to complete the From E $\rightarrow$ D/C before starting this extension):

- **Explore semi-supervised learning to incorporate unlabelled data when labelled training data is limited**

  Try decreasing the percentage of the labelled training data used during fine-tuning - all the training data, 50%, 10%, 1%. Keep a record of the drop in performance as the percentage of training data used is dropped. In this extension you can explore whether it is possible to use only a small percentage of the training data with their labels plus the rest of the training data without their labels which should be considered as the unlabelled training data set and get the same performance as when you use the full labelled training dataset during fine-tuning. Using both labelled and unlabelled training datasets is known as semi-supervised learning and there has been an explosion of methods and approaches to learn under this regime. The following blog gives a very extensive and readable overview: Learning with not Enough Data Part 1: Semi-Supervised Learning. For your project you are, of course, only expected to explore one approach or paper such as one focusing on

  - Some form of consistency regularization or
  - Pseudo-labelling
  - etc...

- There are pre-trained Vision Transformers (VITs) available for example at Hugging Face Vision Transformer (ViT) and descriptions of how to fine-tune at this blog Fine-Tune ViT for Image Classification with HuggingFace Transformers. You can build on this example with the basic project in mind. The report could also highlight the advantages/disadvantages and the level of difficulty of fine-tuning a ResNet.

- **Explore LoRA layers**

  As the available pre-trained models become bigger and bigger, it is becoming more difficult w.r.t. compute and memory to fine-tune them with labelled data for specific applications. Thus even if you have labelled data for a particular problem it can be difficult to use it to adapt a large pre-trained model. One clever recent idea to overcome this problem is LoRA: Low-Rank Adaptation of Large Language Models. For your project you could explore adding LoRA layers to the classification pre-trained model from the first part of the project. Check out Implementing LoRA from Scratch for some tips. It might be more informative to choose a bigger version of the network you original tried (but it is still computational feasible to work with). Grading here will be based on an accurate implementation of the approach as well as an informative and thoughtful set of experiments (data used for training, appropriate ablations, etc) performed to explore the benefits of LoRAs.

- **Masked fine-tuning**

  Another very recent upgrade to fine-tuning within the regime of parameter-efficient fine-tuning is: https://arxiv.org/pdf/2312.10136.pdf. Here only a very small subset of the network's parameters are selected to be updated during the fine-tuning process. A simple but apparently effective idea is used to make this selection. A mask is applied at the parameter update step to zero out the non-selected parameters so the computational effort is the same as normal fine-tuning, but the final empirical results are better. You could explore this idea by building on the first part of the project. Grading here will be based on an accurate implementation of the approach as well as an informative and thoughtful set of experiments (data used for training, appropriate ablations, etc) performed to explore the benefits of the parameter selection process over a couple of simple baseline selection methods.

- **Compress you fine-tuned network**

  Once you have pre-trained network well trained to the new dataset. Investigate if you can extract a lower memory and/or compute version of this fine-tuned network and still maintain a similar level of performance. The most common approaches to compression and lower complexity is to either prune the network weights and/or quantize the activations/weights of the network. Simple weight magnitude pruning is described in Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding by Song Han, Huizi Mao, and William J Dally, ICLR, 2016.

You are, of course, encouraged to come up with your own extensions. But do remember to get them vetted through your project proposal. If you do go for an extension from an E to an A then most of the project report should be devoted to the extension as opposed to the basic project. For the basic assignment you should report the main results and put the more extensive fine-tuning results in the appendix.