# COMP3420 — AI for Text and Vision

Week 04 Lecture 1: Advanced Convolutional Networks for Image Classification

Diego Mollá

COMP3420 2023H1

**Abstract**

This lecture will extend what we have seen about convolutional networks. We will learn about common variants to convolutional layers, and we will look at some of the most popular successful architectures. We will also see an introduction to pre-training and fine-tuning, which is a very popular way to leverage networks that have been trained with different training data, often for a different image classification task.

**Update March 11, 2023**

## Contents



## Reading

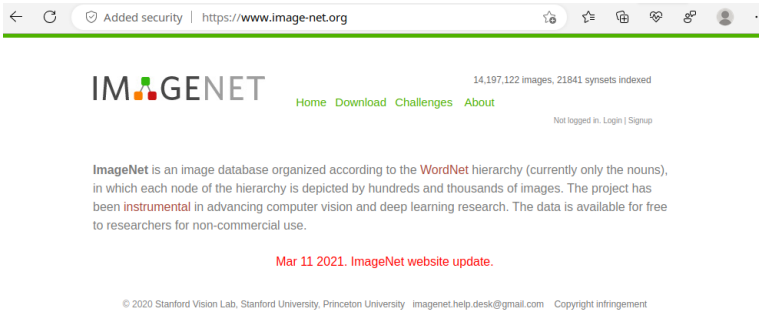
- Deep Learning with Python, 2nd Edition, Chapters 8 & 9.
- Practical Machine Learning for Computer Vision, Chapter 3.

## 1 Pre-training and Fine-Tuning

**Using a Pre-trained Model**

- Many developers release their trained models, ready for re-use.
- These models are typically trained on large amounts of images.
  - A popular image classification dataset: imagenet
- We can re-use these trained models and adapt them to our task.
- By re-using trained models, we can get very good results even with small amounts of training data.

### Imagenet



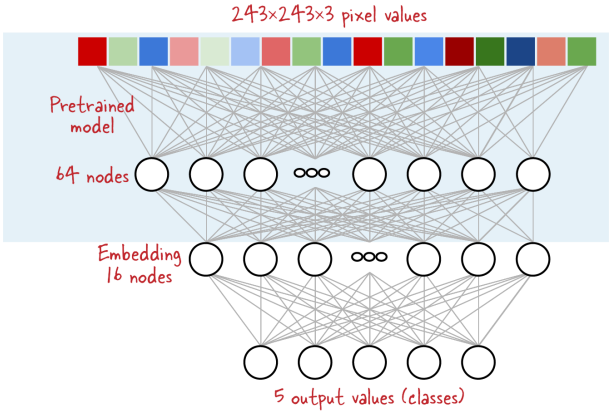

(`https://www.image-net.org/`)

## 1.1 Pretrained Embeddings

### Image Embeddings

- Most of the architecture of a neural networks for image classification can be reused for a different image classification task.
- Recall that the lower-level layers focus on identifying localised patterns and textures.
- These are useful for virtually any classification task.
- In practice, we only need to replace the final classification layer.
- We can say that the goal of a NN for image classification is to obtain the image embeddings, which is then processed by the classification layer.

### Image Embeddings




(Figure 3-2 of Computer Vision book)

Figure 3-2 of "Practical Machine Learning for Computer Vision": The 16 numbers that form the embedding provide a representation of all the information in the entire image.

**Using a Pre-trained Image Embedding**

- When we use a pre-trained image embedding, we normally don't want to modify the pre-trained weights.
- We only update the weights of the final classification layer.
- This way we can avoid the problem of *catastrophic forgetting*.

***Catastrophic Forgetting***

- When we re-train a pre-trained neural network with new data, the information learnt with the new data may override all the information from the previous pre-trained stage.
- We need to find a way for the system not to forget important information learnt during pre-trained.
- But we want the system to be able to adapt to the new task.

**Sample Code**

```
    import tensorflow_hub as hub
    huburl= "https://tfhub.dev/google/imagenet/\
________mobilenet_v2_100_224/feature_vector/4"
    hub.KerasLayer(
        handle=huburl,
        input_shape=(IMG_HEIGHT, IMG_WIDTH, IMG_CHANNELS),
        trainable=False,
        name='mobilenet_embedding')
```

- We prevent catastrophic forgetting by forcing the system not to update the pre-trained weights.
- In Tensorflow Keras, we can use the option trainable=False.

## 1.2 Fine-tuning

**Fine-Tuning**

- Sometimes, we do want to update the pre-trained weights.
- To avoid catastrophic forgetting, usually the learning rate used to update the pre-trained weights is very small.
- Some approaches use differential learning rates, so that learning rates in the upper layers are larger, and learning rates in the lower layers are smaller, or even zero.
- This leverages the fact that, usually, the lower-level layers are useful for most image classification tasks, and higher-level layers are closer to the domain of the task at hand.

**Example: Fine-tuning MobileNet**

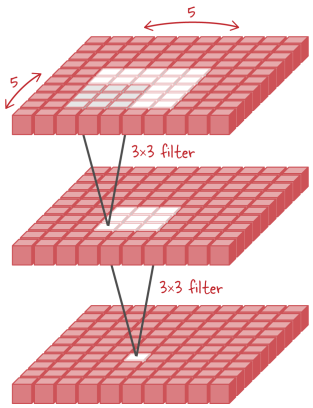
```
  mult_by_layer={
    'block1_': 0.1,
    'block2_': 0.15,
    'block3_': 0.2,
    ... # blocks 4 to 11 here
    'block12_': 0.8,
    'block13_': 0.9,
    'block14_': 0.95,
    'flower_prob': 1.0, # classification head
}

optimizer = AdamW(lr=LR_MAX, model=model,
                  lr_multipliers=mult_by_layer)
```

This example (from the book "Practical Machine Learning for Computer Vision") specifies a maximum learning rate LR_MAX, and a multiplier for each block of layers from MobileNet. Observe that the lower layers have a very small multiplier, and the higher layers have a higher multiplier. The final classification layer (called "classification head") uses the maximum learning rate.

# 2 Extensions to ConvNet

**Deep vs. Wide**




(Figure 3-18 Computer Vision book)

- What is better? A 5 x 5 convolution, or two 3 x 3 convolutions?
- Both options process the same number of pixels (5 x 5).
- But two 3 x 3 convolutions use a lower number of weights.
  - 5 x 5 = 25 weights.
  - 2 x 3 x 3 = 18 weights.

**VGG19**

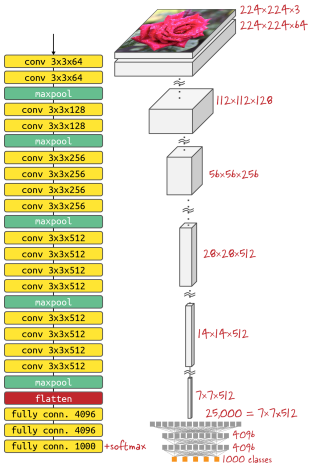
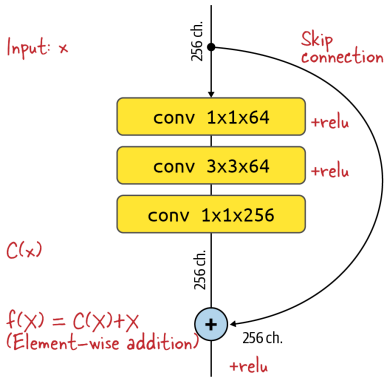



Figure 3-21 of "Practical Machine Learning for Computer Vision". The VGG19 architecture with 19 learnable layers (left). The data shapes are shown on the right (not all represented). Notice that all convolutional layers use 3x3 filters.

### Residual Connections




(Fig 3-27 Computer Vision book)

- A common problem with very deep architectures is that information from lower levels may be lost.
- Residual connections (aka Skip Connections) are shortcuts so that this information is accessible in the higher-level layers.

### The ResNet50 Architecture

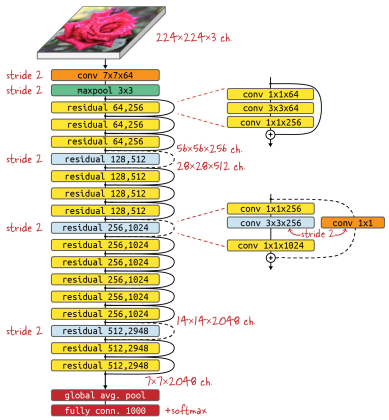



Figure 3-29 of "Practical Machine Learning for Computer Vision". The ResNet50 architecture. Residual blocks with a stride of 2 have a skip connection implemented using a 1x1 convolution (dotted line). The ResNet 101 architecture is similar, with the "residual 256, 1,024" block repeated 23 times instead of 6.

### Take-home Messages

1. Explain the advantages of pre-training and fine-tuning.
2. Explain the various approaches to fine-tune a pre-trained model.
3. Implement an image classification neural network that incorporates a pre-trained model.
4. Explain the use of residual connections.

### What's Next

### Week 5

- Object detection and image segmentation.

### Reading

- Computer Vision book, chapter 4.
- Deep Learning book, chapter 9.