Page 1:

Hello everyone. I am Tanzim Hossain and our project title is Predicting Pulmonary Fibrosis Progression Using Deep Learning.

Page 2:

We want to use transfer learning in our project. Past week we discussed about how we can use transfer learning, approach of it. Today I will be discussing about repurposing of pre-trained model, how to choose pre-trained that is suitable for different problem and architecture of different pre-trained model.

Page 3:

When people are repurposing a pre-trained model for their needs, they start by removing the original classifier, then add a new classifier that fits their purposes, and finally they have to fine-tune their model according to one of three strategies:

A. **Train the entire model.** In this case, people use the architecture of the pre-trained model and train it according to their dataset. If someone learning the model from scratch, so they will need a large dataset and a lot of computational power.

Page 4:

B. **Train some layers and leave the others frozen.** In pre-trained model, lower layers refer to general features which are problem independent, while higher layers refer to specific features which are problem dependent. Here, people can choose how much they want to adjust the weights of the network (a frozen layer does not change during training). Usually, if someone have a small dataset and a large number of parameters, they will leave more layers frozen to avoid overfitting. By contrast, if the dataset is large and the number of parameters is small, we can improve the model by training more layers.

Page 5:

C. **Freeze the convolutional base.** This case corresponds to an extreme situation of the train/freeze trade-off. The main idea is to keep the convolutional base in its original form and then use its outputs to feed the classifier. In this case we are using the pre-trained model as a fixed feature extraction mechanism, which can be useful if someone is short on computational power, dataset is small, and/or pre-trained model solves a problem very similar to the one they want to solve.

Page6:

Unlike Strategy C, whose application is straightforward, Strategy A and Strategy B require to be careful with the learning rate used in the convolutional part. The learning rate is a hyper-parameter that controls how much people adjust the weights of the network. When using a pre-trained model based on CNN, It's always a good practice to use a small learning rate because high learning rates increase the risk of losing previous knowledge. Assuming that

the pre-trained model has been well trained, keeping a small learning rate will ensure that it don't distort the CNN weights too soon and too much.

Page 7:

Classify the problem according to the Size-Similarity Matrix. This matrix classifies problem considering the size of the dataset and its similarity to the dataset in which the pre-trained model was trained. For example, if the task is to identify cats and dogs, ImageNet would be a similar dataset because it has images of cats and dogs. However, if the task is to identify cancer cells, ImageNet can't be considered a similar dataset.

Page 8:

Here we can use the Size-Similarity Matrix to guide our choice and then refer to the three options that were mentioned before about repurposing a pre-trained model.

Quadrant 1. If the dataset falls under quadrant 1 which is the dataset is large, but different from the pre-trained model's dataset. This situation will lead us to Strategy A. Since we have a large dataset, we are able to train a model from scratch and do whatever we want. Despite the dataset dissimilarity, in practice, it can still be useful to initialize our model from a pre-trained model, using its architecture and weights.

Page 9:

Quadrant 2. In case of quadrant 2 which is Large dataset and similar to the pre-trained model's dataset. Any option works. Probably, the most efficient option is Strategy B. Since we have a large dataset, overfitting shouldn't be an issue, so we can learn as much as we want. However, since the datasets are similar, we can save ourselves from a huge training effort by leveraging previous knowledge. Therefore, it should be enough to train the classifier and the top layers of the convolutional base.

Page 10:

Quadrant 3. If the dataset falls under quadrant 3 which is Small dataset and different from the pre-trained model's dataset, everything is against us. In this case the only hope Strategy B. It will be hard to find a balance between the number of layers to train and freeze. If we go deep, our model can overfit, if we stay in the shallow end of our model, we won't learn anything useful. Probably, we will need to go deeper than in Quadrant 2 and we will need to consider data augmentation techniques.

Page11:

Quadrant 4. In case of quadrant 4 which is, dataset is small, but similar to the pre-trained model's dataset, Strategy C is the best option. We just need to remove the last fully-connected layer (output layer), run the pre-trained model as a fixed feature extractor, and then use the resulting features to train a new classifier.

Page 12:

There are perhaps a dozen or more top-performing pre-trained models for image recognition that can be downloaded and used as the basis for image recognition, classification and related tasks. Some of them are as follows:

- VGG (e.g. VGG16 or VGG19).
- GoogleNet (e.g. InceptionV3).
- Residual Network (e.g. ResNet34, ResNet50, ResNet101).

Page 13:

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the Oxford University in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". VGG16 refers to the fact that it has 16 layers that have some weights. 13 convolution and 3 fully connected layer. Total depth is 23 including input, convolution, pooling, fully connected layer and softmax. The dataset used for training, validation and testing is ImageNet dataset. It contains 1.2 million training images, 50,000 validation images, and 150,000 testing images and the size of the images is 256x256.

The input image size of VGG16 is 224x224. Therefore, the images have been down-sampled to a fixed resolution of 224x224. This is a pretty large network, and has a total of about 138.3 million parameters. The model achieves 92.7% test accuracy in ImageNet.

Page 14:

Let's go through the architecture of VGG16.

- The first two layers are convolutional layers with 3×3 filters, and in the first two layers they use 64 filters that end up with a 224×224×64 volume because they are using same convolutions. So, this (CONV64)×2 represents that we have 2conv layers with 64 filters. The filters are always 3×3 with stride of 1 and they're always implemented with the same convolutions.
- Then, they use a pooling layer which will reduce height and width of a volume: it goes from 224×224×64 down to 112×112×64.
- Then, have a couple more conv layers. Here we use 128 filters and because we use the same convolutions, a new dimension will be 112×112×128.
- Then, a pooling layer is added so new dimension will be 56×56×128.
- 2conv layers with 256 filters
- The pooling layer
- A few more conv layers with 512 filters
- A pooling layer
- A few more conv layers with 512 filters
- A pooling layer

- At the end we have final 7×7×512 into Fullyconnected layer (FC) with 4096 units, and in a softmax output one of a 1000 classes. Which results in 138 illion parameters.

Page 15:

VGG19 is similar to VGG1. It has 19 layers that have some weights consist of 16 convolution layer and 3 fully connected layer. The main difference of VGG19 with VGG16 is, it has three more layer of convolution than VGG16. So, Total depth is 26 including input, convolution, pooling, fully connected layer and softmax. For three more layers of convolution Total number of parameters is around 143.6 million. VGG19 which is bigger than VGG16, but because VGG16 does almost as well as the VGG19 a lot of people use VGG16.

Page 16:

Deep networks extract low, middle and high-level features and classifiers in an end-to-end multi-layer fashion, and the number of stacked layers can enrich the "levels" of features. When the deeper network starts to converge, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Such degradation is not caused by overfitting or by adding more layers to a deep network leads to higher training error. The deterioration of training accuracy shows that not all systems are easy to optimize.

Page 17:

To overcome this problem, Microsoft introduced a deep residual learning framework. Instead of hoping every few stacked layers directly fit a desired underlying mapping, they explicitly let these layers fit a residual mapping. The formulation of $F(x)+x$ can be realized by feedforward neural networks with shortcut connections. Shortcut connections are those skipping one or more layers. The shortcut connections perform identity mapping, and their outputs are added to the outputs of the stacked layers. Such residual part receives the input as an amplifier to its output. The dimensions usually are the same. Either way – no additional training parameters are used.

Page 18:

Trained on ImageNet dataset. Input size is 224x224. Total number of parameters of ResNet50 is around 25.6 million, for Rsnet101 is 44.7 million and for ResNet152 is 60.4 million. ResNet50, Resnet101 and resNet152 Achieved an accuracy of 92.1%, 92.8% and 93.1% respectively on the test image.

Page 19:

Deeper network such as ResNet50, ResNet101 use bottleneck layer to improve efficiency. Keeps the time complexity same as the two layered convolution. Which Allows us to increase

the number of layers as well as to converge much faster. 152-layer ResNet has 11.3 billion FLOPS while VGG-16/19 nets has 15.3/19.6 billion FLOPS.

Page 20:

The plain baselines, which is the middle figure are mainly inspired by the philosophy of VGG networks. The convolutional layers mostly have 3×3 filters and follow two simple rules:

Number one is, for the same output feature map, the layers have the same number of filters;

And number two is, if the size of the features map is halved, the number of filters is doubled to preserve the time complexity of each layer.

It is worth noticing that the ResNet model has fewer filters and lower complexity than VGG networks.

Based on the above plain network, a shortcut connection is inserted (right side figure) which turn the network into its counterpart residual version. The identity shortcuts can be directly used when the input and output are of the same dimensions as you can see from the solid line shortcut. When the dimensions increase (dotted line shortcuts) it considers two options:

First is the shortcut performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no additional parameter.

Second is the projection shortcut is used to match dimensions (done by 1×1convolutions).

For either of the options, if the shortcuts go across feature maps of two size, it performed with a stride of 2.

Each ResNet block is either two layers deep (used in small networks like ResNet 34) or 3 layers deep (such as ResNet 50, 101, 152).

That's all. Thank you….