Summary/Review

 # Transfer Learning

 # What is transfer learning?

Most popular models are difficult to train from scratch as they require huge datasets (like ImageNet), a large number of training iterations and very heavy computing machinery. The basic features (edges, shapes) learned by early layers in a network are generalizable. While the later layers in an already trained network tend to capture features that are more particular to a specific image classification task. Transfer learning uses the idea that if we keep the early layers of a pre-trained network, and re-train the later layers on a specific dataset, we might be able to leverage some state of that network on a related task.

 # A typical transfer learning workflow in Keras

1. Initialize base model, and load pre-trained weights (e.g. ImageNet).
2. "Freeze" layers in the base model by setting `training = False`.
3. Define a new model that goes on top of the output of the base model's layers.
4. Train resulting model on your data set.

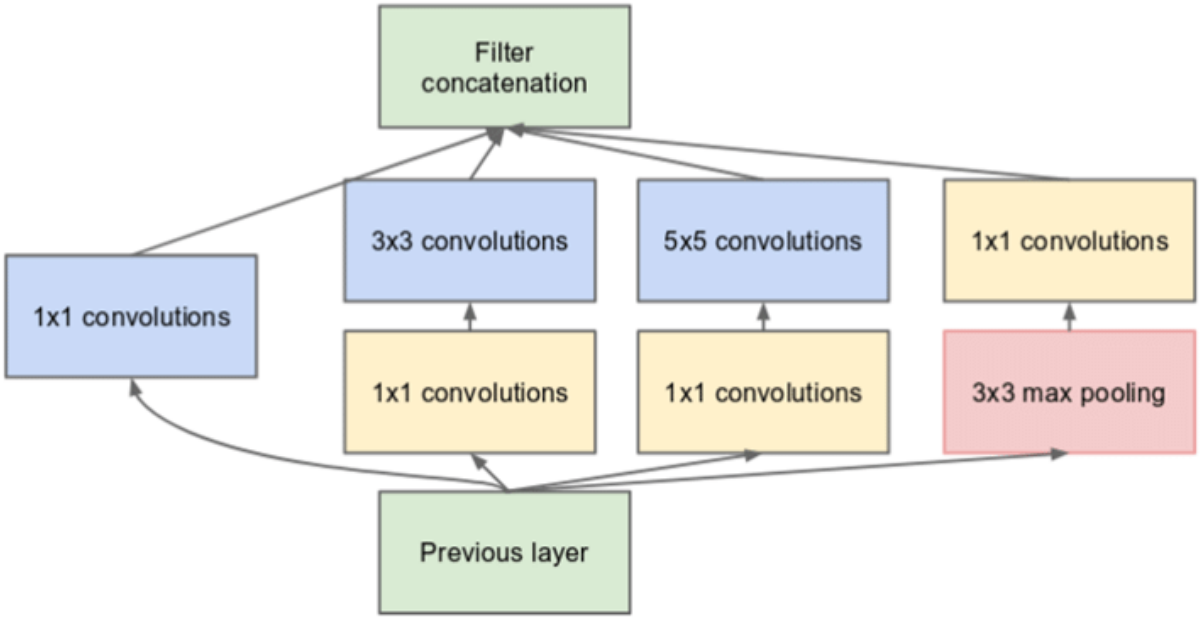
 # Fine tuning in transfer learning

Fine-tuning is an optional step in transfer learning, it usually ends up improving the performance of the model. It is easy to overfit the model in this step as we are re-training the entire model. So we use regularization (dropout layers), a lower learning rate, a small number of epochs (training iterations), and early stopping to know when the model has stopped improving and to prevent overfitting.

 # Modern CNN Architectures

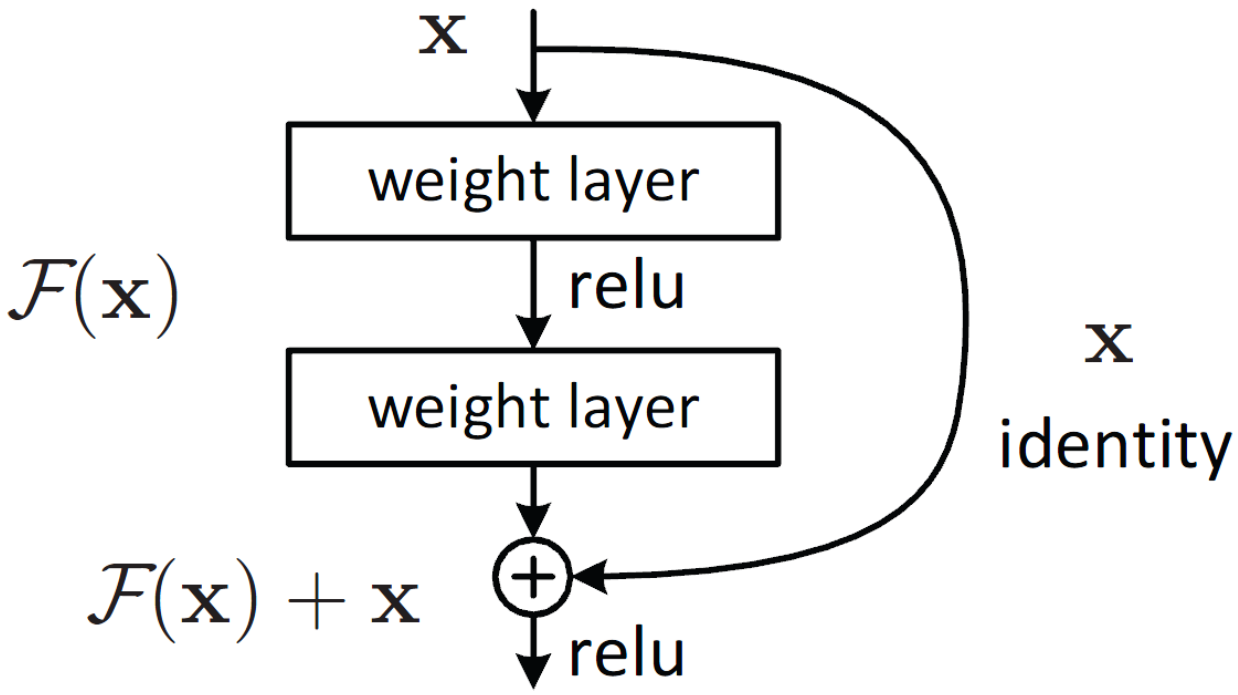
 # Inception - V3

Instead of focusing on increasing the depth of the network, InceptionNet focuses on increasing the width and depth of the model simultaneously to attain better accuracy, while keeping the computing resources constant. It focuses on **parallel processing** and extraction of various feature maps concurrently using **Inception modules**, which are collections of convolutions with different filter sizes and pooling operations. The following is an illustration of the inception module in inception-v1 architecture:



 # ResNet - 50

ResNet features special skip connections which add the output from an earlier layer directly to a later layer and a heavy use of batch normalization. It allows us to design deep CNNs without compromising the model's convergence and accuracy. The basic building blocks for ResNets is the convolution and identity blocks. Essentially, ResNet uses the network layers to fit a residual mapping F(x)+x,$F(x)+x$, instead of trying to learn the desired underlying mapping $H(x)$ directly with stacked layers.



 # Regularization Techniques

- L2 (Ridge) regularization
- L1 (Lasso) regularization
- Dropout
- Batch Normalization
- Data shuffling