Day 5: Data Science Interview Preparation

Q1: What are Epochs?

One Epoch is an ENTIRE dataset is passed forwards and backwards through the neural network.

Since one epoch is too large to feed to the computer at once, we divide it into several smaller batches.

We always use more than one Epoch because *one epoch leads to underfitting.*

As the number of epochs increases, several times the weight are changed in the neural network and the curve goes from underfitting up to optimal to overfitting curve.

Q2. What is the batch size?

Batch Size

The total number of training and examples present in a single batch.

Unlike the learning rate hyperparameter where its value doesn't affect computational time, the batch sizes must be examined in conjunctions with the execution time of training. The batch size is limited by hardware's memory, while the learning rate is not. Leslie recommends using a batch size that fits in hardware's memory and enables using larger learning rate.

If our server has multiple GPUs, the total batch size is the batch size on a GPU multiplied by the numbers of GPU.

If the architectures are small or your hardware permits very large batch sizes, then you might compare the performance of different batch sizes.

Also, recall that small batch sizes add regularization while large batch sizes add less, so utilize this while balancing the proper amount of regularization. It is often better to use large batch sizes so a larger learning rate can be used.

Q3: What is dropout in Neural network?

Dropout refers to ignoring units during the training phase of a certain set of neurons which is chosen randomly. These units are not considered during the particular forward or backward pass.

More technically, at each training stage, individual nodes are either dropped out of the net with probability *1-p* or kept with probability *p*, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.

We need Dropout *to prevent over-fitting.*

A dropout is an approach to regularization in neural networks which helps to reduce interdependent learning amongst the neurons.

Where to use

Dropout is implemented per-layer in a neural network.

It can be used with most types of layers, such as dense fully connected layers, convolutional layers, and recurrent layers such as the long short-term memory network layer.

Dropout may be implemented on any or all hidden layers in the network as well as the visible or input layer. It is not used on the output layer.

Benefits:

1. Dropout forces a neural network to learn more robust features that are very useful in conjunction with different random subsets of the other neurons.

2. Dropout generally doubles the number of iterations required to converge. However, the training time for each epoch is less.

Q4: List down hyperparameter tuning in deep learning.

The process of setting the hyper-parameters requires expertise and extensive trial and error. There are no simple and easy ways to set hyper-parameters — specifically, learning rate, batch size, momentum, and weight decay.

Approaches to searching for the best configuration:

- Grid Search
- Random Search

Approach

1. Observe and understand the clues available during training by monitoring validation/test loss early in training, tune your architecture and hyper-parameters with short runs of a few epochs.

2. Signs of *underfitting* or *overfitting* of the test or validation loss early in the training process are useful for tuning the hyper-parameters**.**

Tools for Optimizing Hyperparameters

- Sage Maker
- Comet.ml
- Weights &Biases
- Deep Cognition
- Azure ML

Q5: What do you understand by activation function and error functions?

Error functions

In most learning networks, an error is calculated as the difference between the predicted output and the actual output.

$$J(w) = p - \widehat{p}$$

The function that is used to compute this error is known as Loss Function J(.). Different loss functions will give different errors for the same prediction, and thus have a considerable effect on the performance of the model. One of the most widely used loss function is mean square error, which calculates the square of the difference between the actual values and predicted value. Different loss functions are used to deals with a different type of tasks, i.e. regression and classification.

Regressive loss functions:
Mean Square Error
Absolute error
Smooth Absolute Error

Classification loss functions:

1.Binary Cross-Entropy
2.Negative Log-Likelihood
3.Margin Classifier
4.Soft Margin Classifier

Activation functions decide whether a neuron should be activated or not by calculating a weighted sum and adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

In a neural network, we would update the weights and biases of the neurons based on the error at the outputs. This process is known as back-propagation. Activation function makes the back-propagation possible since the gradients are supplied along with the errors to update the weights and biases.

Q6: Why do we need Non-linear activation functions?

A neural network without activation functions is essentially a linear regression model. The activation functions do the non-linear transformation to the input, making it capable of learning and performing more complex tasks.

1. Identity
2. Binary Step
3. Sigmoid
4. Tanh
5. ReLU
6. Leaky ReLU
7. Softmax

The activation functions do the non-linear transformation to the input, making it capable of learning and performing more complex tasks.

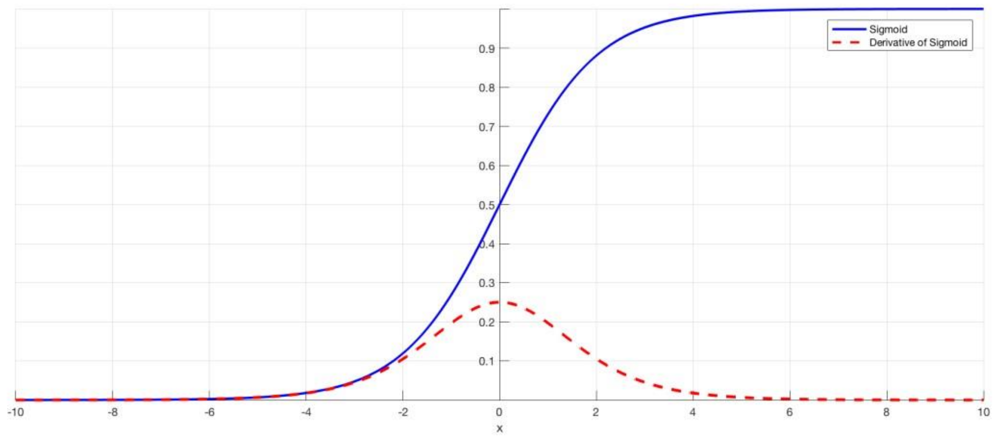
Q7: What do you under by vanishing gradient problem and how can Do we solve that?

The problem:

As more layers using certain activation function are added to neural networks, the gradients of the loss function approach zero, making the networks tougher to train.

Why:

Certain activation functions, like the sigmoid function, squishes a large input space into a small input space between 0 and 1. Therefore, a large change in the input of the sigmoid function will cause a small change in the output. Hence, the derivative becomes small.



For shallow networks with only a few layers that use these activations, this isn’t a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively.

However, when *n* hidden layers use an activation like the sigmoid function, *n* small derivatives are multiplied together. Thus, the gradient decreases exponentially as we propagate down to the initial layers.

Solutions:
The simplest solution is to use other activation functions, such as ReLU, which doesn't cause a small derivative.

Residual networks are another solution, as they provide residual connections straight to earlier layers.

Finally, batch normalization layers can also resolve the issue.

Q8: What is Transfer learning in deep learning ?

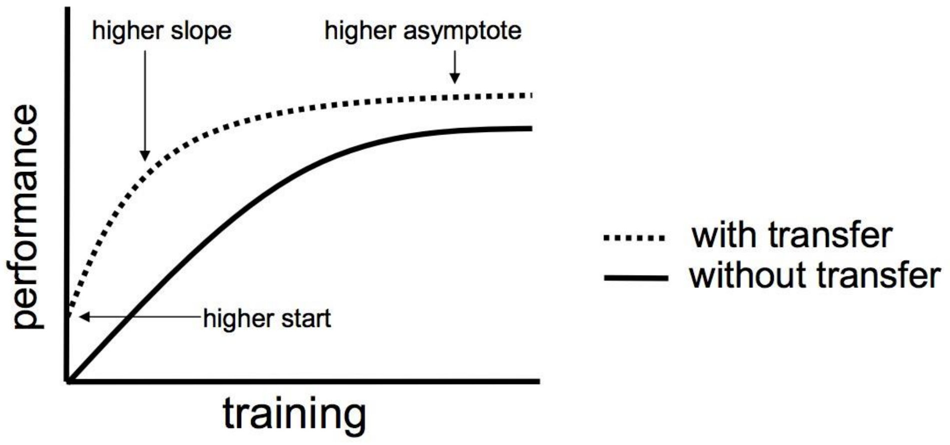
Transfer learning: It is a machine learning method where a model is developed for the task is again used as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems

Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.

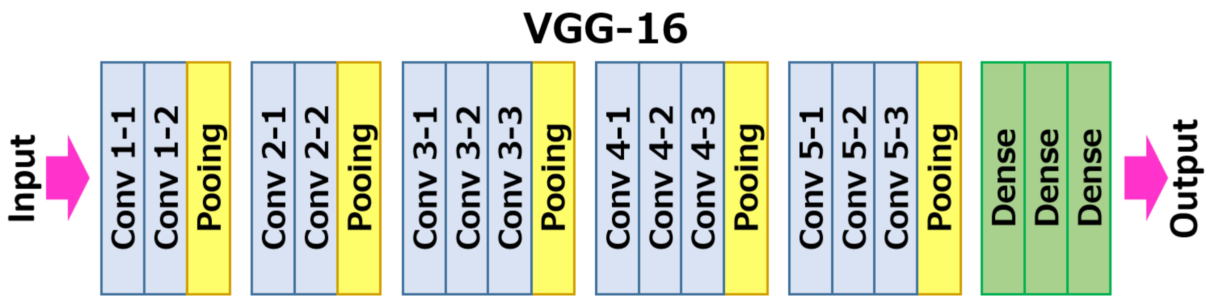
Transfer learning is an optimization that allows rapid progress or improved performance when modelling the second task.

Transfer learning only works in deep learning if the model features learned from the first task are general.



Q9: What is VGG16 and explain the architecture of VGG16?

VGG-16 is a simpler architecture model since it's not using many hyperparameters. It always uses 3 x 3 filters with the stride of 1 in convolution layer and uses SAME padding in pooling layers 2 x 2 with a stride of 2.

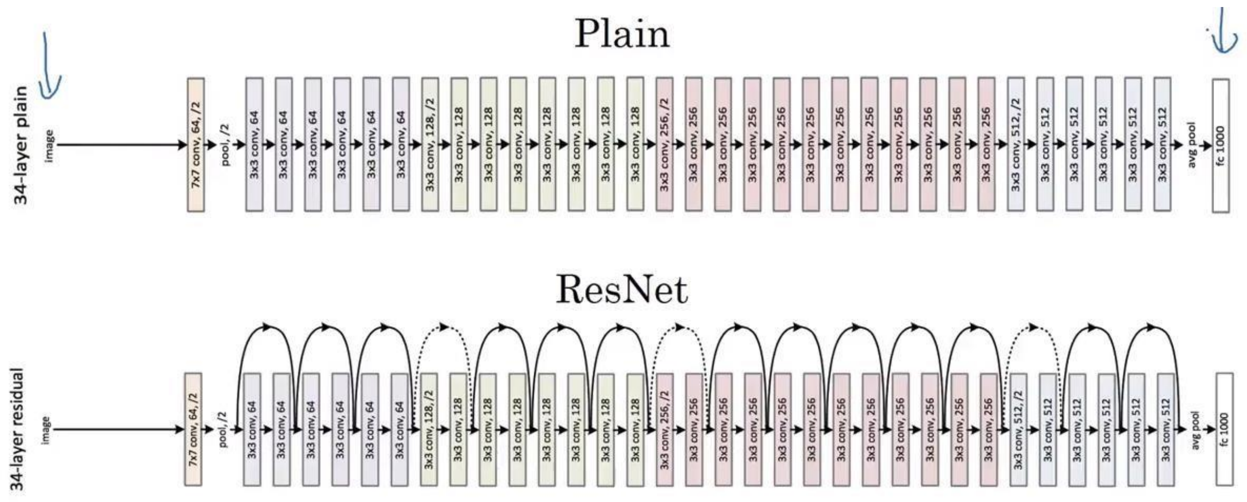


This architecture is from the VGG group, Oxford. It improves AlexNet by replacing the large kernelsized filter with multiple 3X3 kernel-sized filters one after another. With a given receptive field(the effective area size of input image on which output depends), multiple stacked smaller size kernel is better than the one with a larger size kernel because multiple non-linear layers increases the depth of the network which enables it to learn more complex features, and that too at a lower cost.

Three fully connected layers follow the VGG convolutional layers. The width of the networks starts at the small value of 64 and increases by a factor of 2 after every sub-sampling/pooling layer. It achieves the top-5 accuracy of 92.3 % on ImageNet.

Q10: What is RESNET?

The winner of ILSRVC 2015, it also called as Residual Neural Network (ResNet) by Kaiming. This architecture introduced a concept called "skip connections". Typically, the input matrix calculates in two linear transformations with ReLU activation function. In Residual network, it directly copies the input matrix to the second transformation output and sums the output in final ReLU function.



Skip Connection

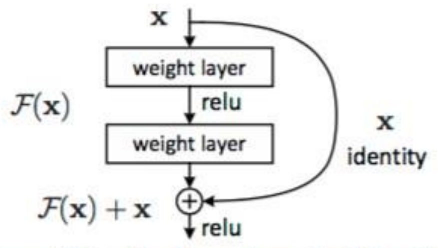

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2. Residual learning: a building block.

Experiments in paper four can judge the power of the residual network. The plain 34 layer network had high validation error than the 18 layers plain network. This is where we realize the degradation problems. And the same 34 layers network when converted to the residual network has much less training error than the 18 layers residual network.

Q11: What is ImageNet?

ImageNet is a project aimed at (manually) labelling and categorizing images into almost 22,000 separate object categories for computer vision researches.

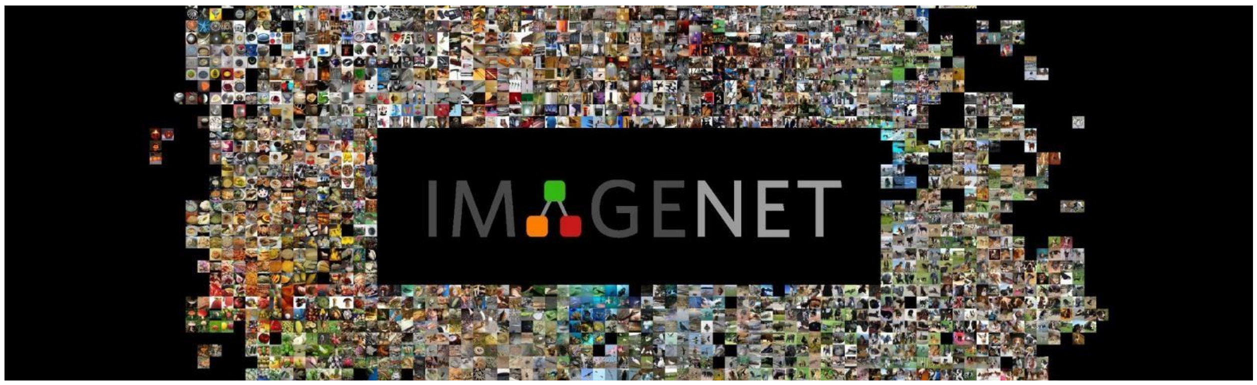
When we hear the about*"ImageNet"* in the context of deep learning and Convolutional Neural Network, we are referring to *ImageNet Large Scale Visual Recognition Challenge*.

The main aim of this image classification challenge is to train the model that can correctly classify an input image into the 1,000 separate objects category.

Models are trained on the ~1.2 million training images with another 50,000 images for validation and 100,000 images for testing.

These 1,000 image categories represent object classes that we encounter in our day-to-day lives, such as species of dogs, cats, various household objects, vehicle types, and much more.

When it comes to the image classification, the ImageNet challenge is the “de facto “ benchmark for computer vision classification algorithms — and the leaderboard for this challenge has been dominated by Convolutional Neural Networks and Deep learning techniques since 2012.



Q12: What is DarkNet?

DarkNet is a framework used to train neural networks; it is open source and written in C/CUDA and serves as the basis for YOLO. Darknet is also used as the framework for training YOLO, meaning it sets the architecture of the network.

Clone the repo locally, and you have it. To compile it, run a make. But first, if you intend to use the GPU capability, you need to edit the **Makefile** in the first two lines, where you tell it to compile for GPU usage with CUDA drivers.

Q13: What is YOLO and explain the architecture of YOLO (you only Look Once). One use case?

YOLO v1

The first YOLO You only look once (YOLO) version came about May 2016 and sets the core of the algorithm, the following versions are improvements that fix some drawbacks.

In short, YOLO is a network “inspired by” Google Net. It has 24 convolutional layers working as the feature extractors and two dense layers for making the predictions. The architecture works upon is called Darknet, a neural network framework created by the first author of the YOLO paper.

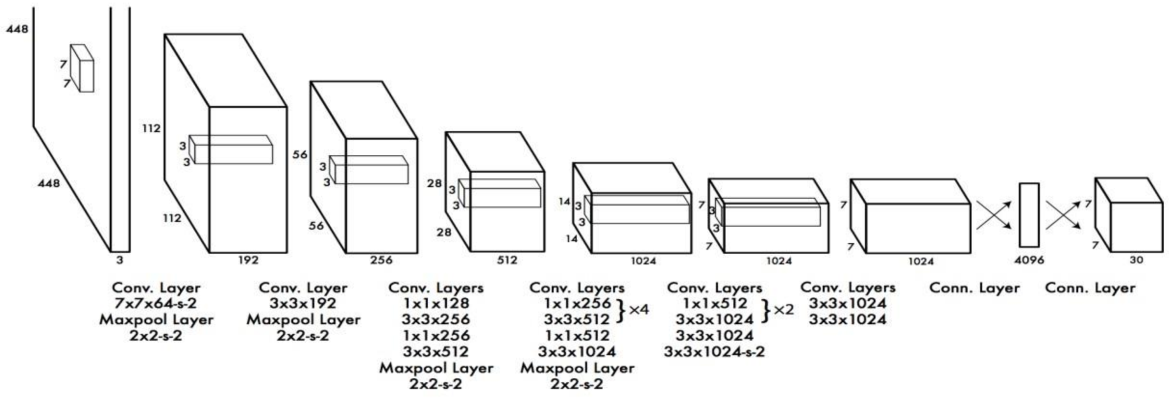
Core Concept:

The algorithm works off by dividing the image into the grid of the cells, for each cell bounding boxes and their scores are predicted, alongside class probabilities. The confidence is given in terms of IOU (*intersection over union*), metric, which is measuring how much the detected object overlaps with the ground truth as a fraction of the total area spanned by the two together (the union).

YOLO v2:

This improves on some of the shortcomings of the first version, namely the fact that it is not very good at detecting objects that are very near and tends to make some of the mistakes on localization.

It introduces a few newer things: Which are *anchor boxes* (pre-determined sets of boxes such that the network moves from predicting the bounding boxes to predicting the offsets from these) and the use of features that are more fine-grained so smaller objects can be predicted better.



YOLO v3-

YOLOv3 came about April 2018, and it adds small improvements, including the fact that bounding boxes get predicted at the different scales. The underlying meaty part of the YOLO network, Darknet, is expanded in this version to have 53 convolutional layers.