# Advanced CNN Architecture

- Save the model
- Model Prediction
- Simplest Project (MNIST Data)
- Image Classification task

## Advanced CNN Architecture

- AlexNet → ImageNet (RGB (3 channels))
  - 1000 unique classes

```
## Layer 1: Conv + MaxPooling
 model_alexnet.add(layers.Conv2D(96, (11, 11), strides=(4,4), activation='relu',
input_shape=input_shape))
 model_alexnet.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

 ## Layer 2: Conv + MaxPooling
 model_alexnet.add(layers.Conv2D(256, (5,5), padding='same', activation='relu'))
 model_alexnet.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))

 ## Layer 3: Conv
 model_alexnet.add(layers.Conv2D(384, (3,3), padding='same', activation='relu'))

 ## Layer 4: Conv
 model_alexnet.add(layers.Conv2D(384, (3,3), padding='same', activation='relu'))

 ## Layer 5: Conv + MaxPooling
 model_alexnet.add(layers.Conv2D(256, (3,3), padding='same', activation='relu'))
 model_alexnet.add(layers.MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
```

Conv + Maxpooling

Conv

Conv + Maxpooling

## Neural Network Architecture

```
## Flatten and Dense Layers
model_alexnet.add(layers.Flatten())
model_alexnet.add(layers.Dense(4096, activation='relu'))
## to avoid overfitting
model_alexnet.add(layers.Dropout(0.5))
model_alexnet.add(layers.Dense(4096, activation='relu'))
## to avoid overfitting
model_alexnet.add(layers.Dropout(0.5))
```

Input layer

Hidden Layer

Dropout → Reduce overfitting

↑ Error = ↑ Bias + variance ↑

Low Bias & High variance

What might be the possible reasons of Overfitting ?? (DL context)

Early Stopping ← Stops the training when Validation set Performance Stops improving

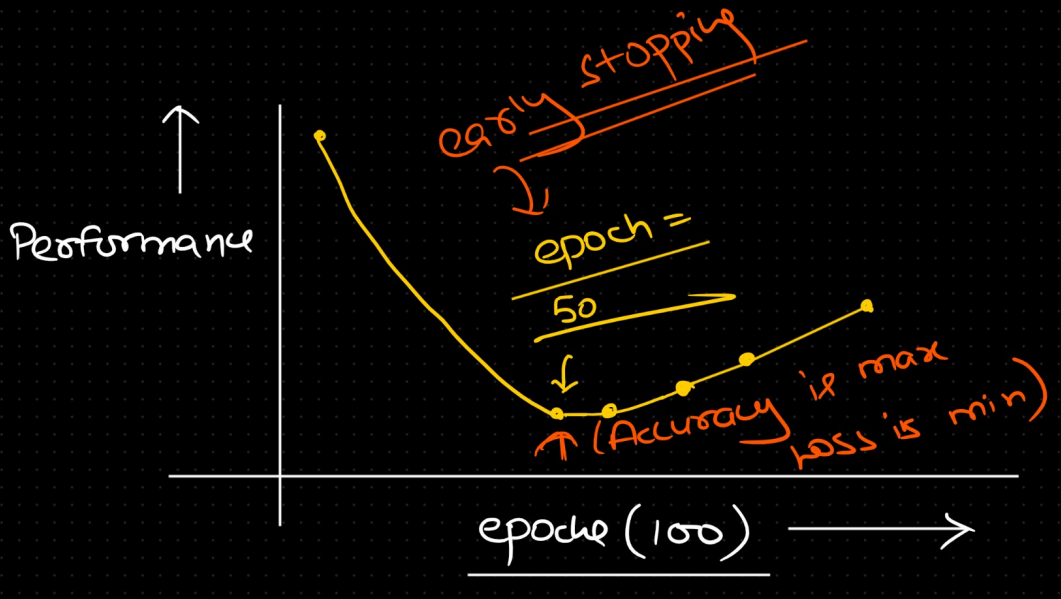
1. Too much training
2. Insufficient training data
   - ↳ Data Augmentation ← (flipping, rotation, cropping of similar original images) (Healthcare Domain)
3. Noisy or irrelevant data
4. Lack of Regularisation (L1, L2 (Lasso, Ridge))

Dropout

↓

Randomly deactivate neurons during training

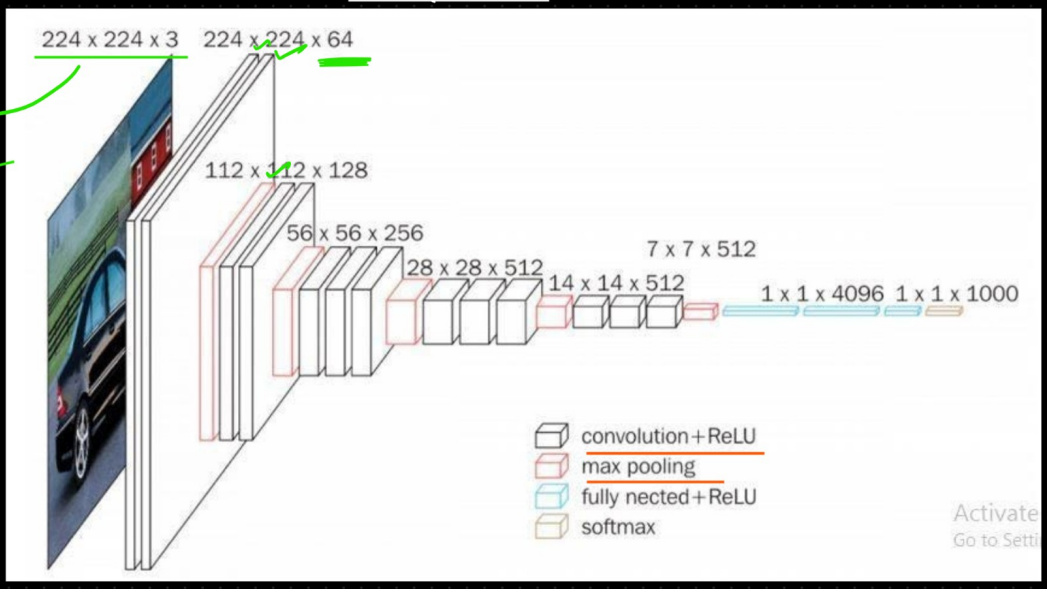
5. Poor Hyperparameter tuning (Learning Rate, Batch Size)



VGG 16 → ImageNet

↳ simpler architecture

VGG 19




ImageNet

→ Convolutional Layer

↳ filter size = 3 x 3

stride = 1

→ Padding = 'same' (input & output image dimensions are same)

Standardize the values

→ MaxPooling Layer

↳ filter = 2 x 2

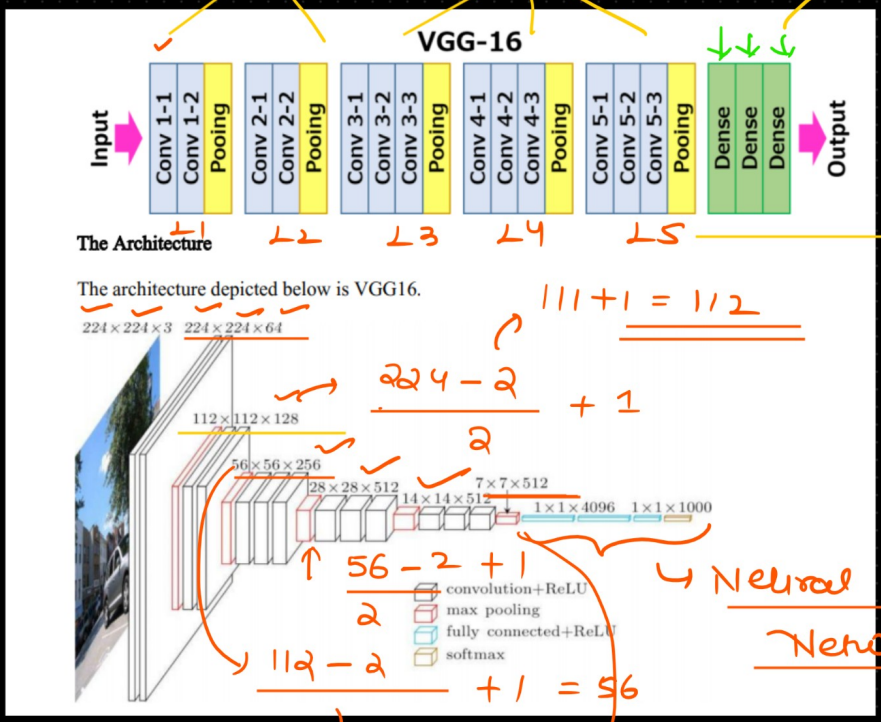
stride = 2

$$\Rightarrow \left\lfloor \frac{n - f + 2p}{s} \right\rfloor + 1$$

2 Conv + 1 MaxPooling

3 Conv + 1 Maxpooling

Neural Network




111 + 1 = 112

$\frac{224 - 2}{2} + 1$

$\frac{56 - 2}{2} + 1$

$\frac{112 - 2}{2} + 1 = 56$

Neural Network

Input Value

25088 → Hidden layer

4096

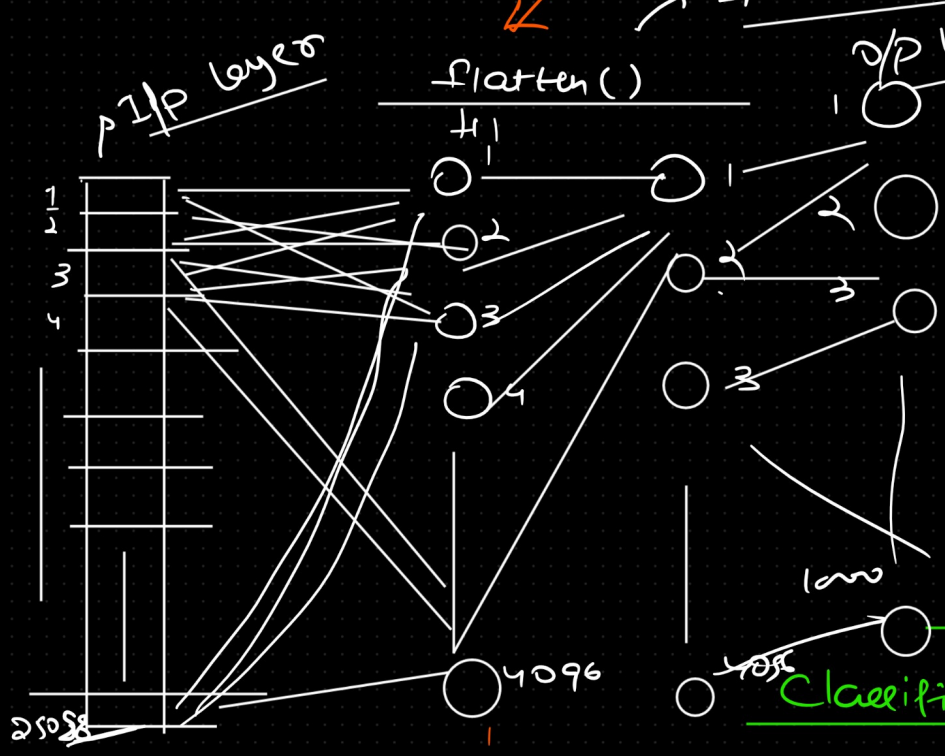
4096

Output layer

1000

1000 Classes

ImageNet Classification task



very huge (# neurons)
↳ training time huge
computationally expensive task

very less (# neurons)
↳ overfitting

↳ better

| Feature | AlexNet (2012) | VGG16 (2014) |
|---|---|---|
| Total Parameters | ~60M | ~138M |
| Convolution Layers | 5 | 13 (VGG16), 16 (VGG19) |
| Fully Connected Layers | 3 | 3 |
| Kernel Sizes | 11x11, 5x5, 3x3 | Only 3x3 |
| Max Pooling | 3x3, stride 2 | 2x2, stride 2 |
| Activation Function | ReLU | ReLU |
| Dropout | Yes | Yes |
| Final Classifier | Softmax (1000 classes) | Softmax (1000 classes) |

VGG16
↳ 13
3

VGG19
↳ 16
3

VGG16 better than AlexNet → More layers

↳ easier to understand & implement

↳ better in terms of performance

extract more complex & hierarchical features

better Performance

in recognizing fine details in an image

# Transfer Learning

RESNET → Task to explore & do some research on your own.