One Shared Neuron

Ken Ko & Muhammad Tanveer

Outline

- Stage 3
  - Baseline
  - VGGNet Inspired
  - Inc Convolutional Layers & Filters
  - Adam Optimizer
  - Takeaways from Stage 3
- Stage 4
  - Baseline: He Normal Initializer
  - Skipped Connections
  - Accuracy Stagnation
  - LeakyRELU + Data Augmentation
- Stage 5
  - Max Pooling Layers
  - Where to Go From Here?
  - More Filters + Swish + Data Augmentation
  - Depthwise Separable Conv
  - Final Model

Slide Presenters

Stage 3

- Baseline (Mo)
- Model 3 (Ken)
- Model 4 (Mo)
- Model 5 (Mo)
- Takeaways (Ken)

Stage 4

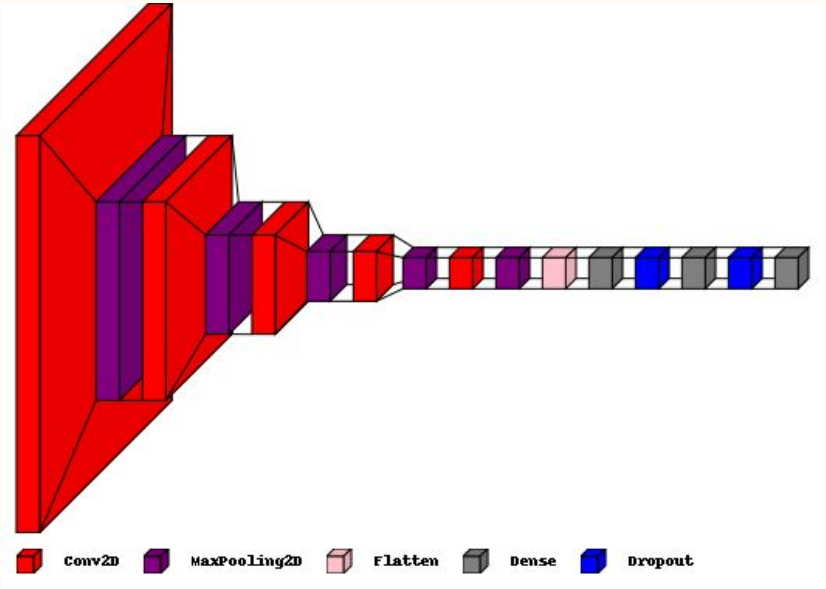
- Baseline (Mo)
- Model 6 (Ken)
- Model 7 (Ken)
- Model 9 (Mo)

Stage 5

- Baseline Model 10 (Ken)
- Model 13 (Ken)
- Model 17 (Mo)
- Model 20 (Ken)

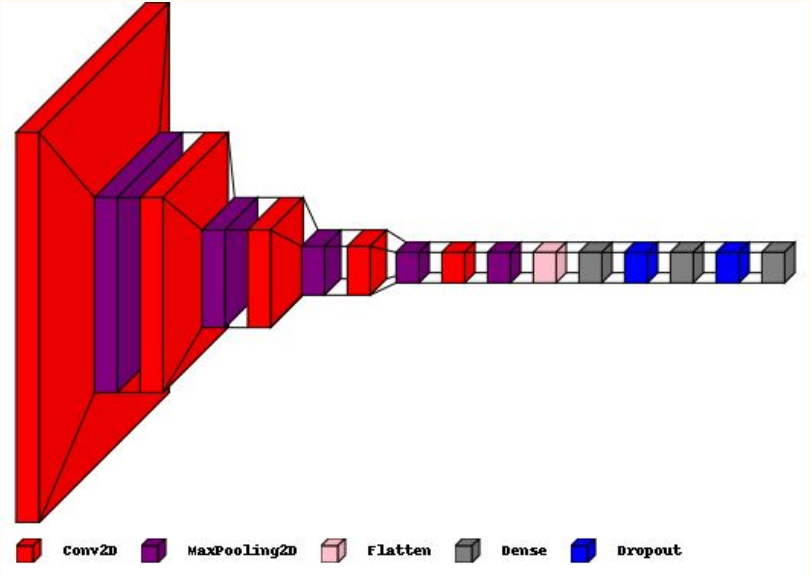
Stage 3: Baseline Model

- Started with the model used in Lab Assignment 7, which had a 92% accuracy on MNIST dataset
- The baseline model attained a test accuracy of 56%



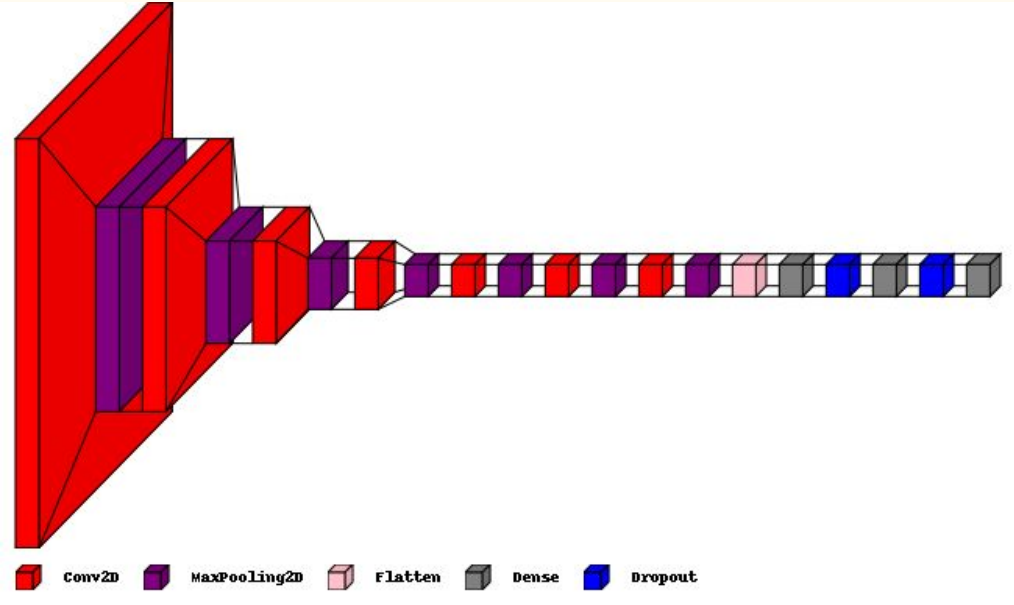
Model 3: VGGNet led to overfitting

- Inspiration: VGGNet
- One Dense Layer with 4096 units
- Resulted in overfitting to training set
- 97% train accuracy, 72% test accuracy



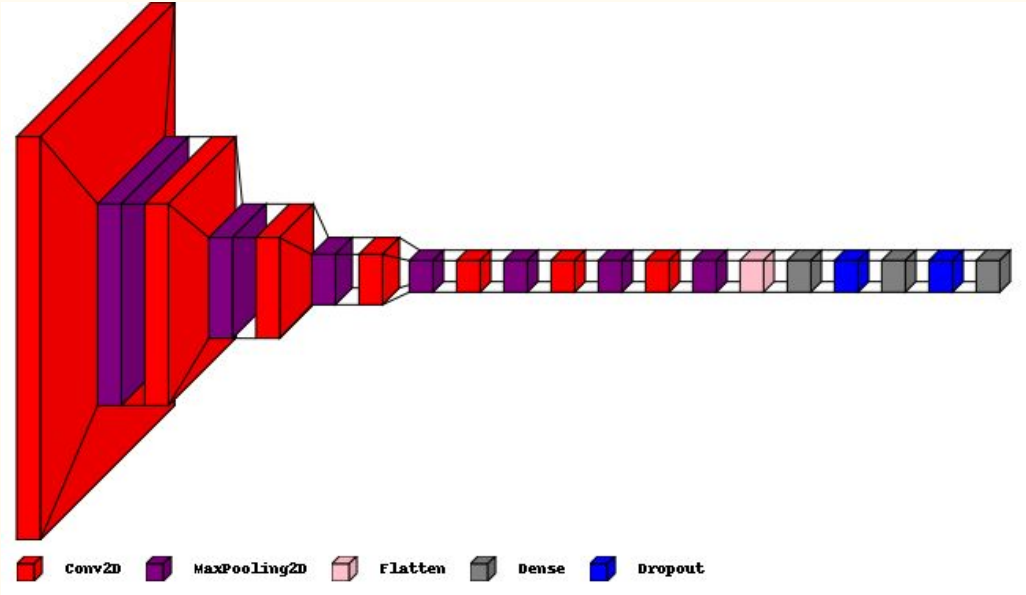
Model 4: Inc Conv Layers & Filters

- Combined model using increased Conv layers and filter size
- Based on results from previous model experiment
- Test accuracy increased to 74%



Model 5: Adam Optimizer

- Optimizer changed from nadam to adam
- Still using previous model architecture
- Increased test accuracy to 74%

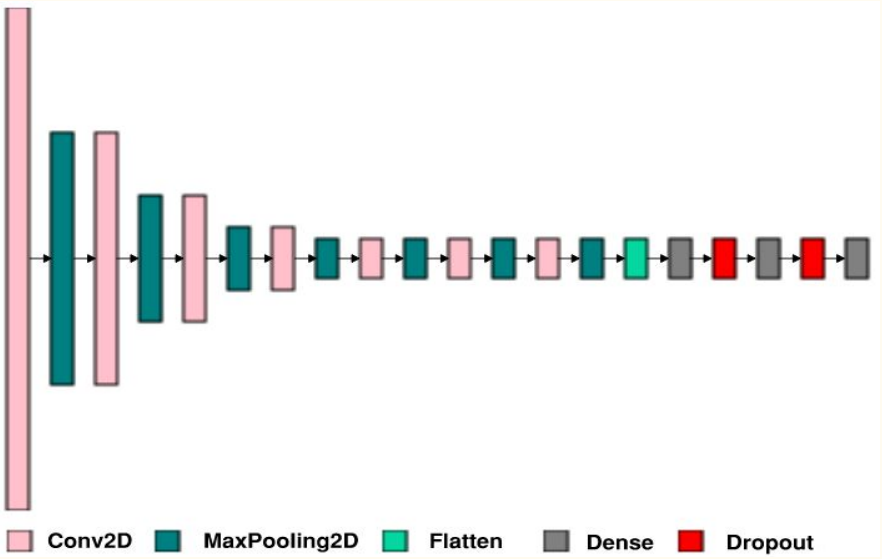


Takeaways from Stage 3

- Issues with unzipping and reading the data
- Train time of ~15 mins/epoch
- Notebook crashes
- Class Mode: Sparse vs Categorical vs Binary

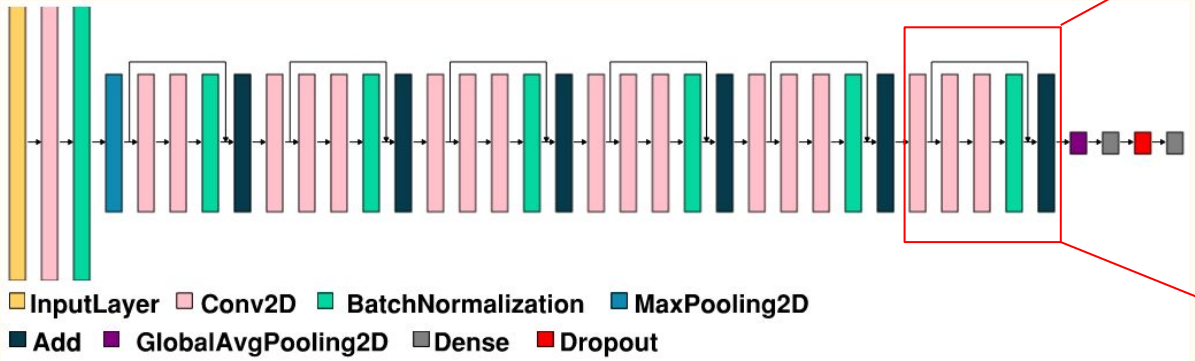
Stage 4 Baseline: He Normal Initialization

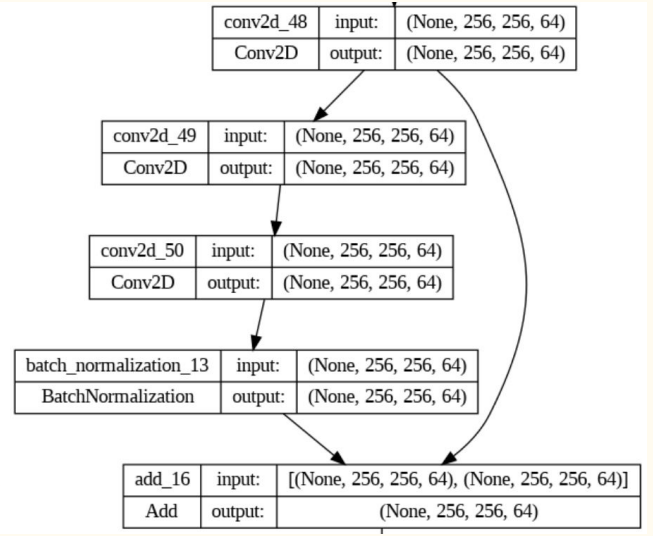
- Changed default Glorot to He Normal weight initialization
- Using same architecture as model submitted for Stage 3
- Accuracy increased to 80%



Model 6: Introducing Skipped Connections

- Skipped Connections with constant Convolution filters
- Accuracy began to stagnate at 20%
- Implemented Batch Normalization to fix issue





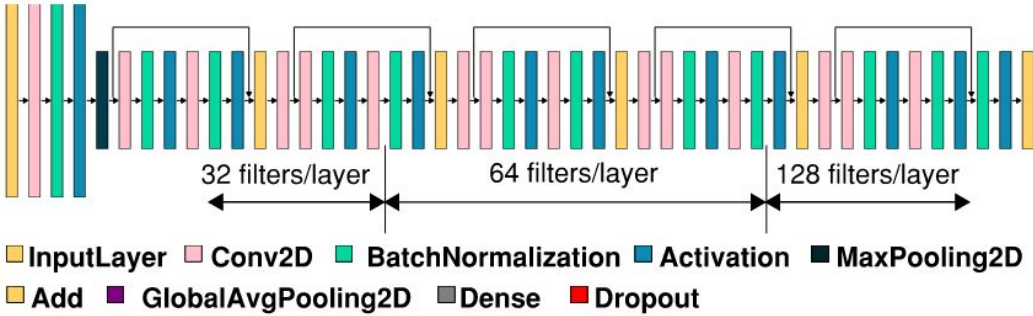
Model 6: Introducing Skipped Connections

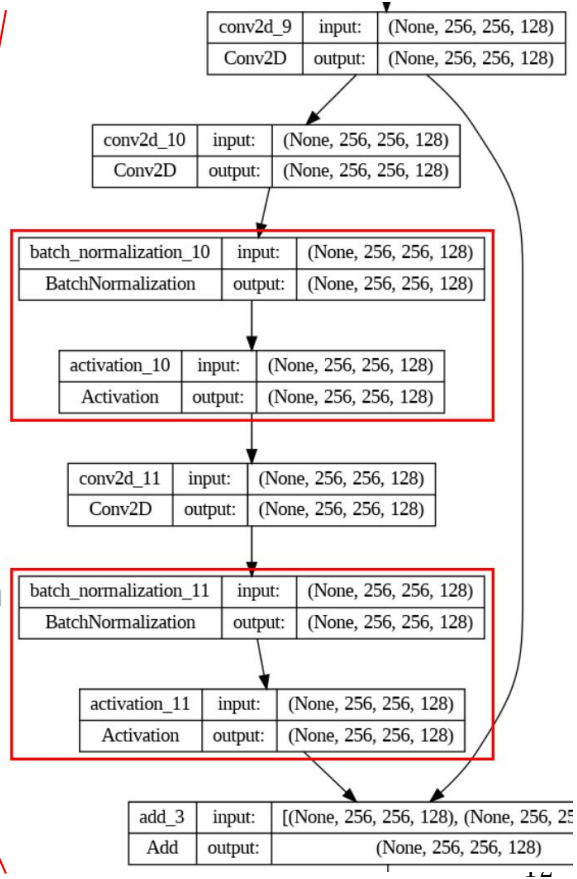
Results:

Architecture	M5	M6	Change
Test accuracy	0.8004	0.8089	+0.8%
Val accuracy	0.7951	0.8133	+2%
Train accuracy	0.8108	0.8501	+4%
Epochs	30	29	-1

Model 7: Accuracy Stagnation

- Based on resNet and VGGNet, filters were increased alongside depth
- Issue 1: Accuracy stagnated at 20%
- Issue 2: Required more RAM than available to run





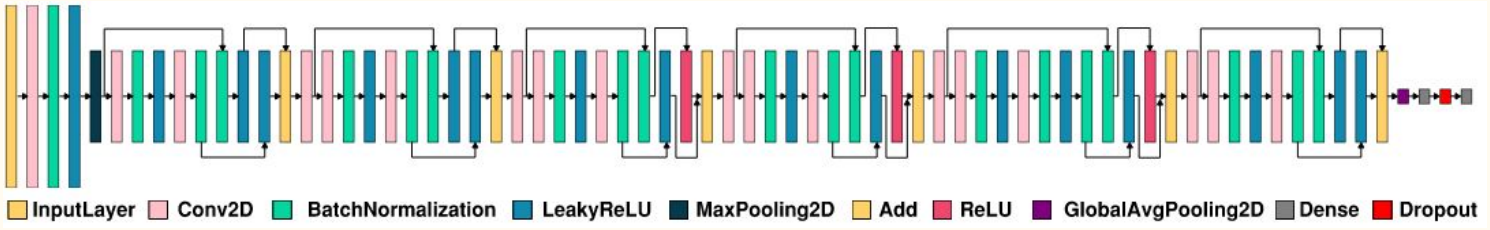
Model 7: Accuracy Stagnation

Results: Overall decrease in performance, almost double the parameters

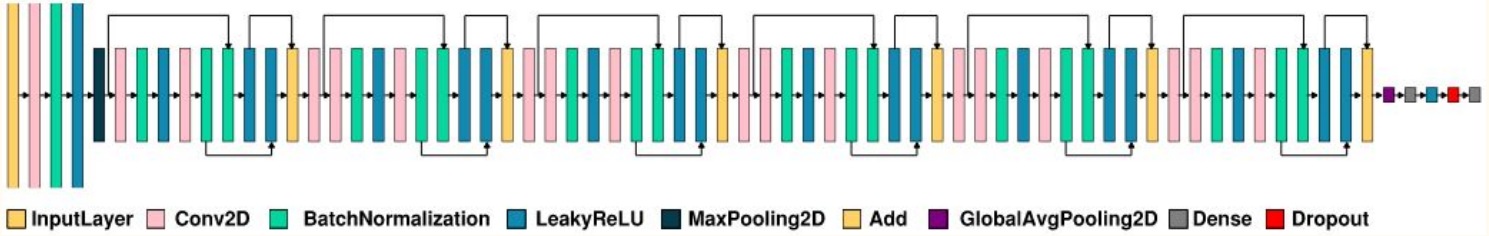
Architecture	M6	M7	Change
Test accuracy	0.8089	0.7289	-8%
Val accuracy	0.8133	0.7231	-9%
Train accuracy	0.8501	0.7696	-8%
Epochs	29	24	-5

Model 9: LeakyRELU + Data Augmentation

- Trained using Data Augmentation
- Used LeakyRELU & He Init from previous model
- Model architecture had mistakes were corrected



Model 8



Model 9

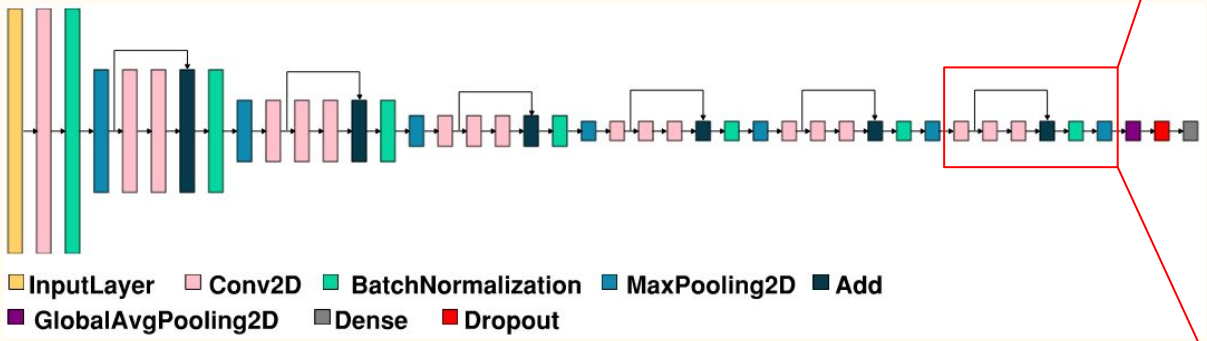
Model 9: LeakyRELU + Data Augmentation

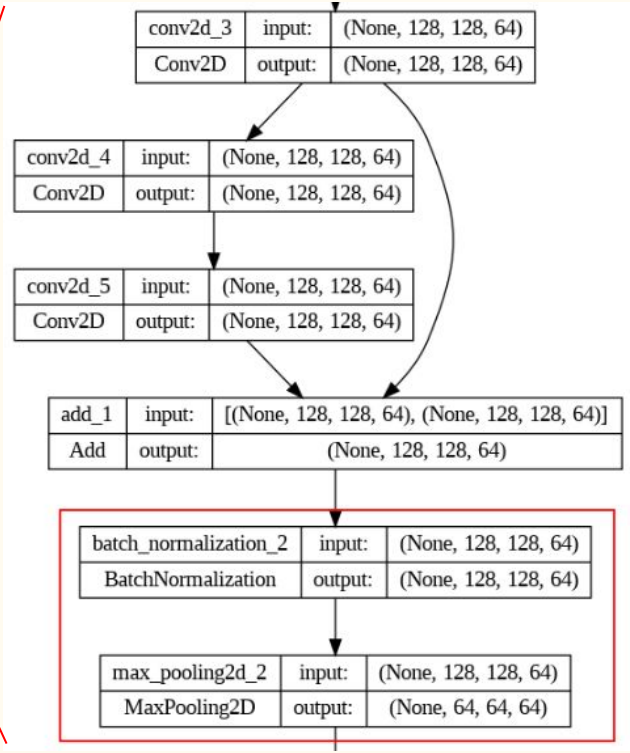
Results: Performance not as good as M6

Architecture	M6	M9	Change
Test accuracy	0.8089	0.6427	-16%
Val accuracy	0.8133	0.6360	-18%
Train accuracy	0.8501	0.7467	-10%
Epochs	29	30	+1

Stage 5: Model 10: Architecture Improvements

- Added MaxPooling layers
- Reduced model complexity





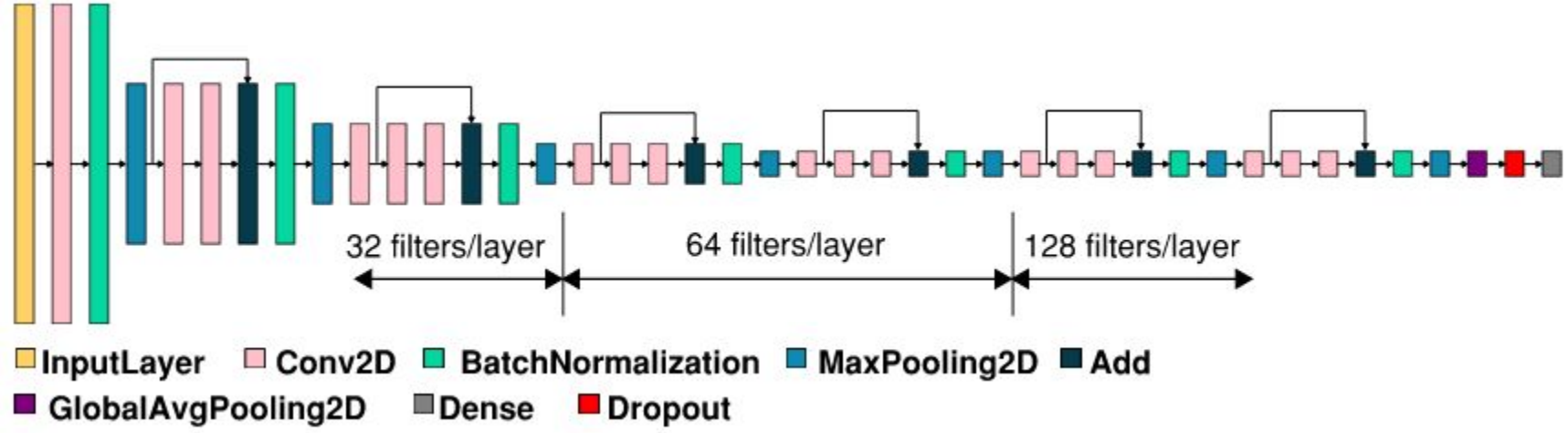
Stage 5: Model 10

Results: Overall improvement but overfits as training progresses

Architecture	Best of Stage 4	M10	Change
Test accuracy	0.8089	0.8564	+5%
Val accuracy	0.8133	0.8440	+3%
Train accuracy	0.8501	0.8899	+4%
Epochs	29	20	-9

Model 13

- Added more filters + swish + data augmentation



Model 13

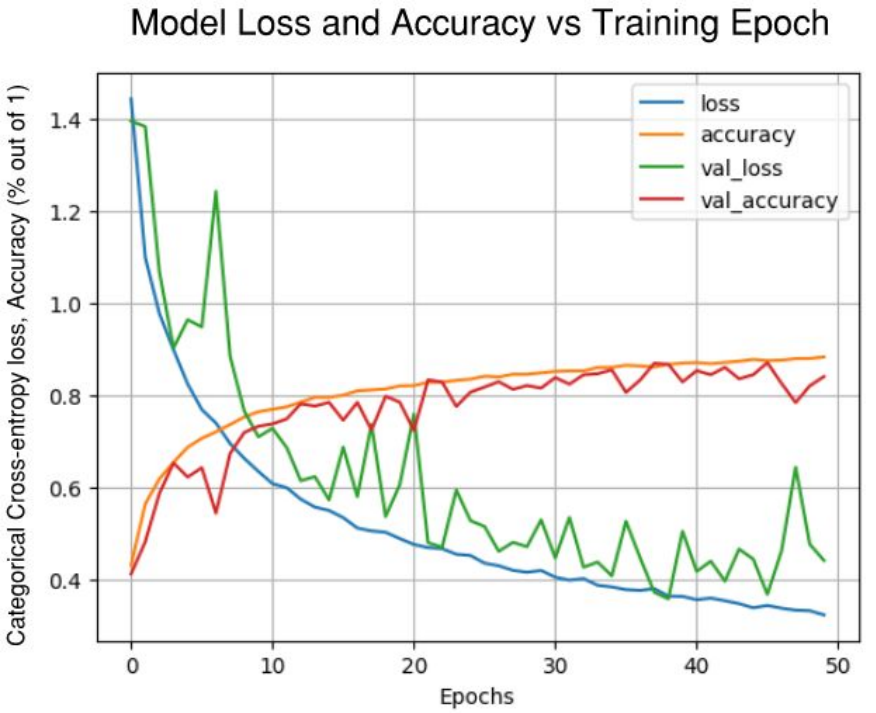
Results: Eliminates overfitting; long training times

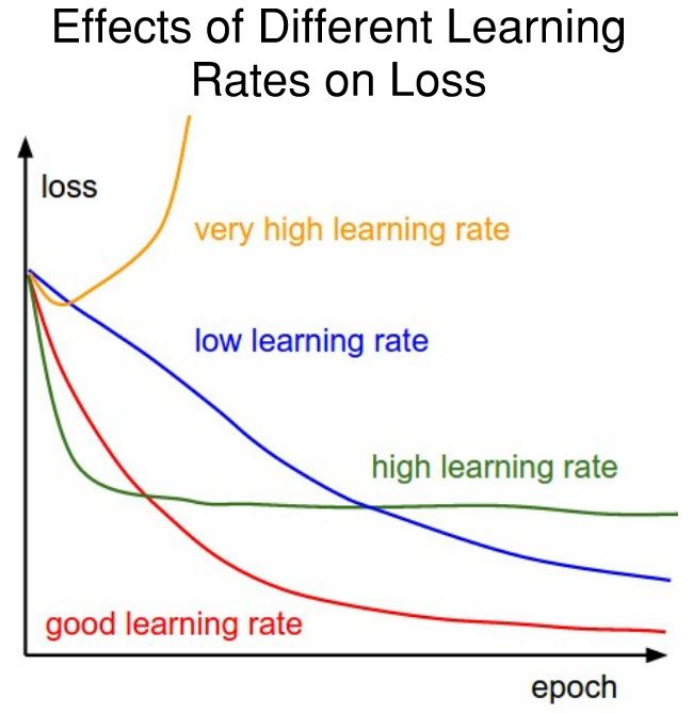
Architecture	M10	M13	Change
Test accuracy	0.8564	0.8787	+2%
Val accuracy	0.8440	0.8911	+5%
Train accuracy	0.8899	0.9124	+2%
Epochs	20	55	+35
Training time	2 hrs	26 hrs	+24 hrs

Where To Go From Here?

- Adjust learning rate?
- Try depth-wise separable convolutions for time savings?
- Add more layers for underfitting?
- Train on full dataset

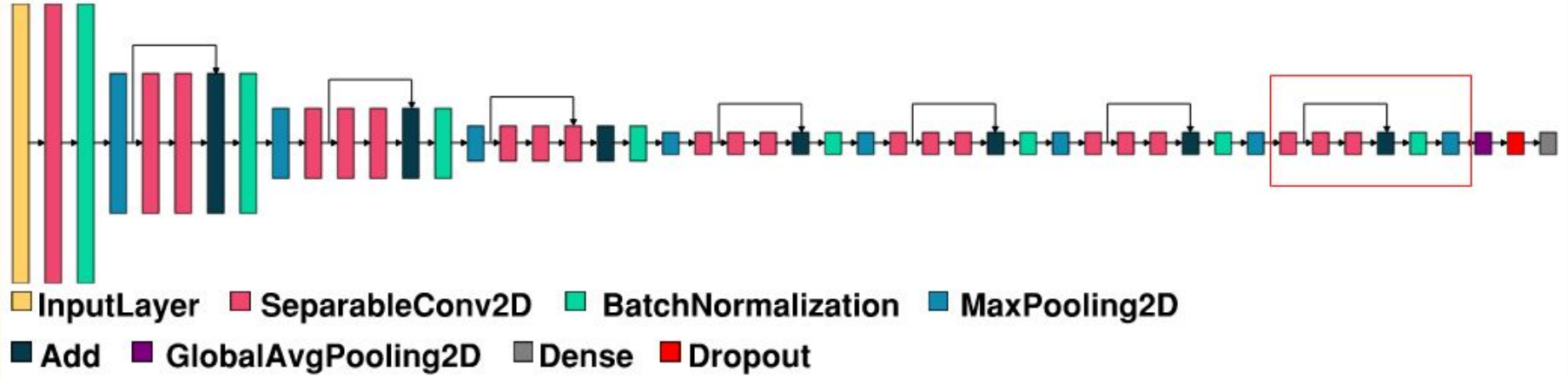
Learning Rate Hyperparameter





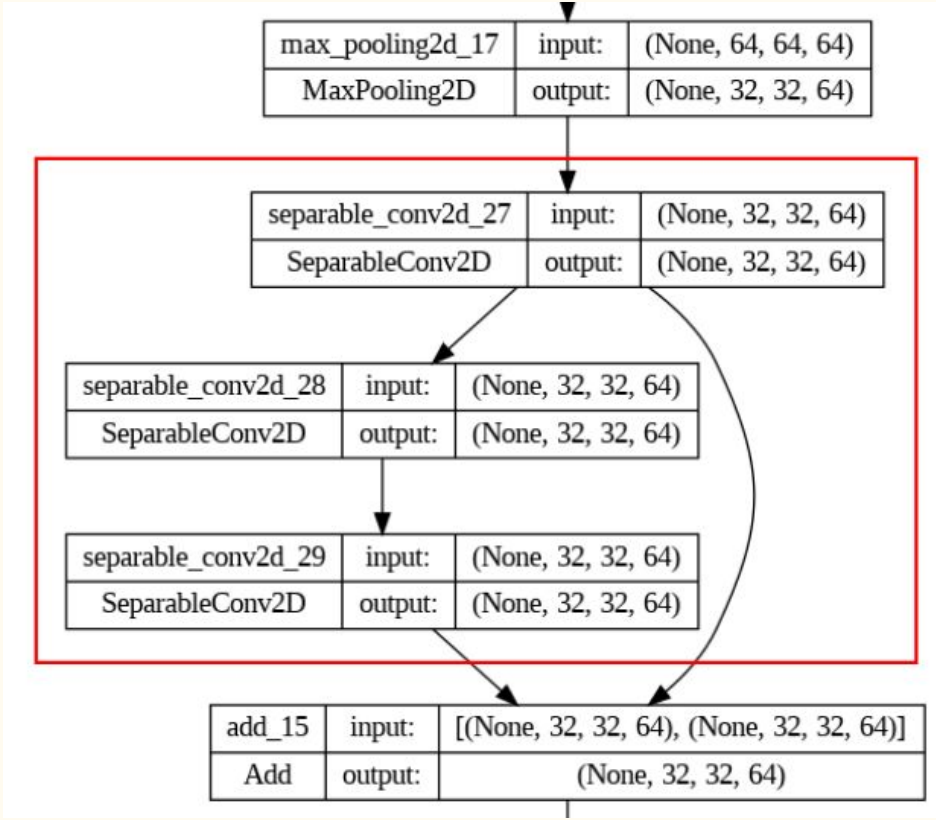
Model 17: Layers + SeparableConv2D

- Added 3 convolutional layers + normalization + maxpooling layers
- Changed Conv2D layers to SeparableConv2D



Model 17

- Changed Conv2D layers to SeparableConv2D



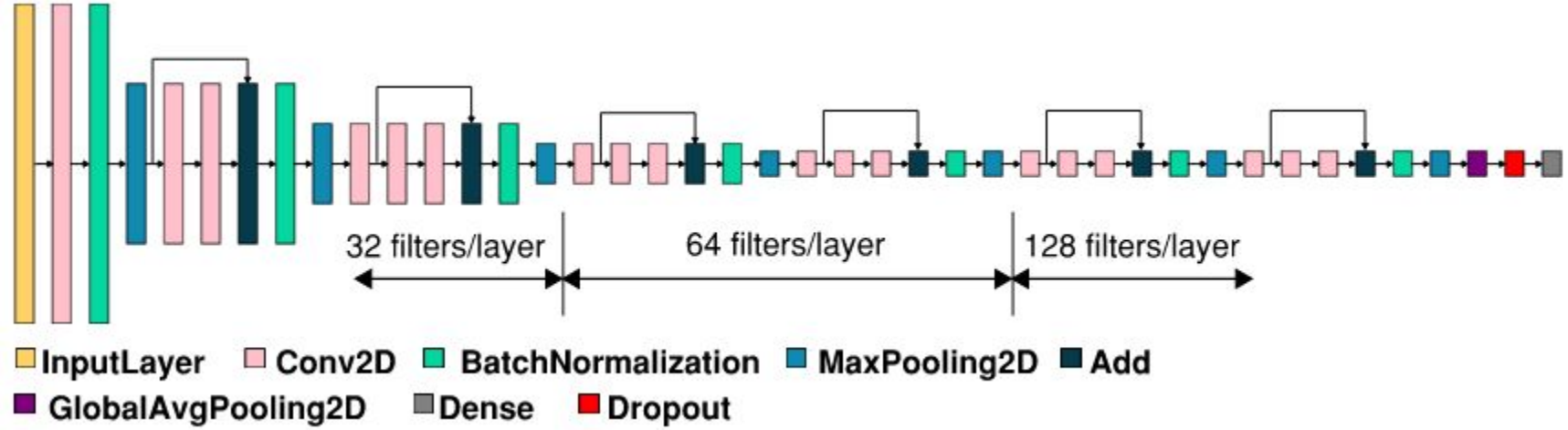
Model 17

Results: Overall decrease in performance, no decrease in training time

Architecture	M13	M17	Change
Test accuracy	0.8787	0.8111	-6%
Val accuracy	0.8911	0.8040	-9%
Train accuracy	0.9124	0.8706	-4%
Epochs	55	19	-36

Final Model 20 Detailed

Change: Trained on full dataset using 90/10 train/val split



```
image_gen = ImageDataGenerator
            (fill_mode='nearest',
             rescale=1./255.,
             rotation_range=45,
             width_shift_range=0.05,
             height_shift_range=0.05,
             shear_range=0.05,
             zoom_range=0.1,
             horizontal_flip=True,
             vertical_flip=False,
             brightness_range=[0.2,1.2]
             )
```



```
inputs = Input(shape=(512,512,3))

# Add the preprocessing/augmentation layers.
x = RandomRotation(0.45)(inputs)
x = RandomTranslation((-0.05, 0.05),(-0.05, 0.05))(x)
x = RandomZoom(.1, .1)(x)
x = RandomFlip(mode='horizontal')(x)
x = RandomBrightness(0.2, value_range=(0.0,1.0))(x)

x = Conv2D(filters=32, kernel_size=5, activation='swi
           kernel_initializer = he_normal(seed = choi
```

Final Model 20 Detailed

Results:

Architecture	M13	M20	Change
Test accuracy	0.8787	0.9000*	+2%
Val accuracy	0.8911	0.8858	-0.5%
Train accuracy	0.9124	0.9168	+0.4%
Epochs	55	86	+31
Training time	26 hrs	13 hrs	-13 hrs

Thank You!