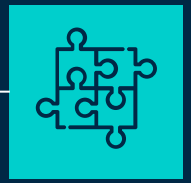
Quick! Draw! Doodle Recognition

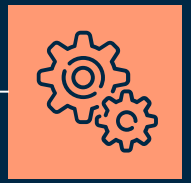
By Marcus Tan

TABLE OF CONTENTS



01

Problem Statement
& EDA



02

CNN model &
Transfer
learning

03

Conclusion
&
Recommendations

Problem Statement & EDA

01

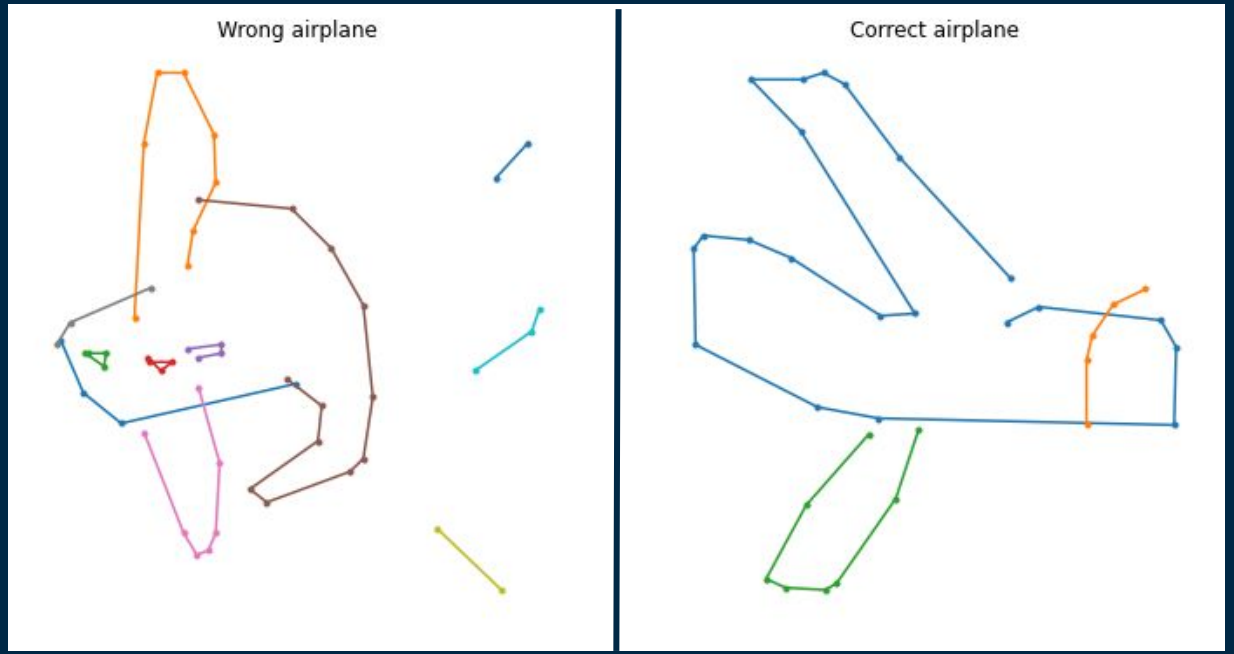
Problem Statement

- Dataset of over 50 million hand drawn drawings with 340 labels
- Tasked by Google to build a better classifier
- Challenge is to predict the drawings within 3 guesses
- Metric - Accuracy & Top 3 accuracy
- Immediate impact on handwriting recognition applications (e.g. Automatic Speech Recognition)

Datasets

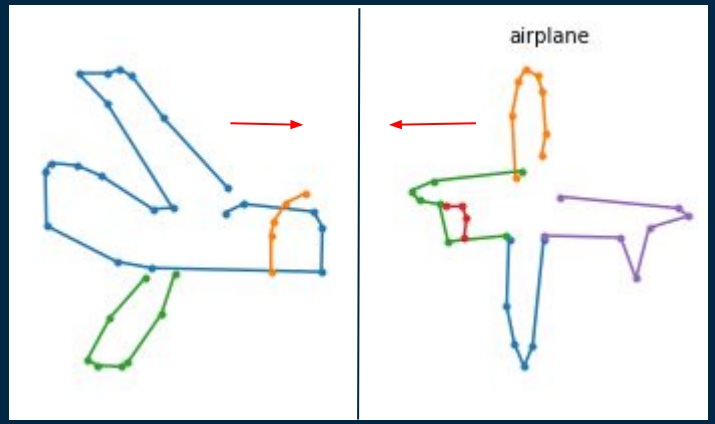
- Train set (340 labels - totalling 50 million+ drawings)
- Test set (340 labels - totalling 100k+ drawings)
- Dataset has been preprocessed by kaggle
- No cleaning needed, No missing values
- Dataset contains drawing strokes, labels and recognized (True/False)

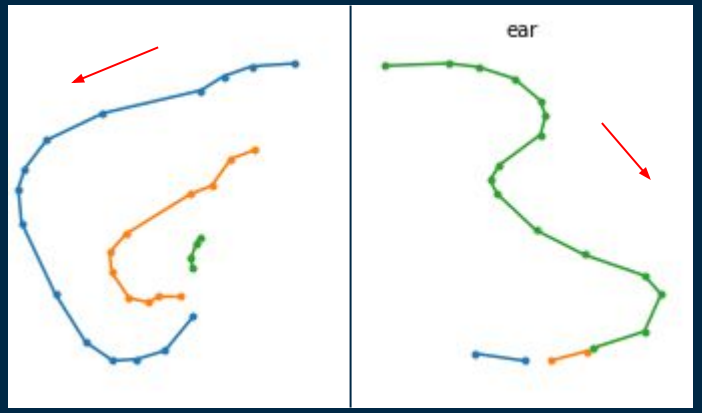
Drawing examples

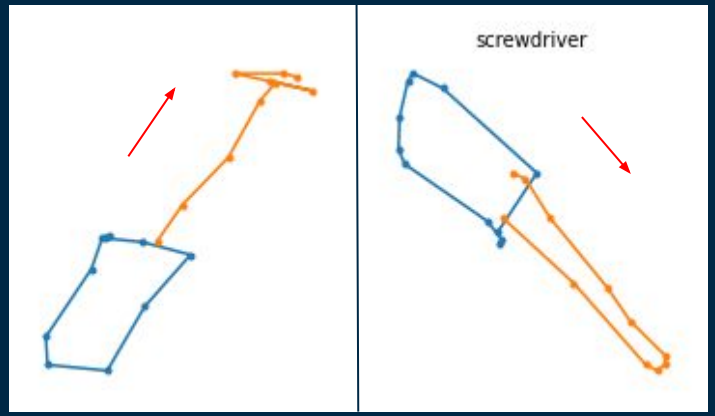


- Left is recognized as False
- Right is recognized as True
- Left one is oddly shaped and random strokes

More drawing examples (True)

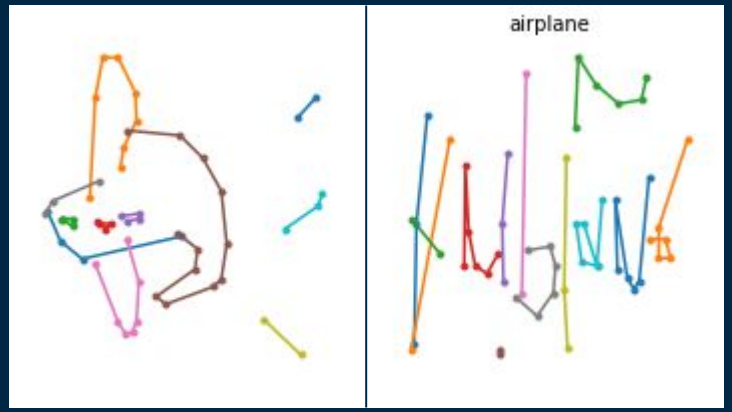


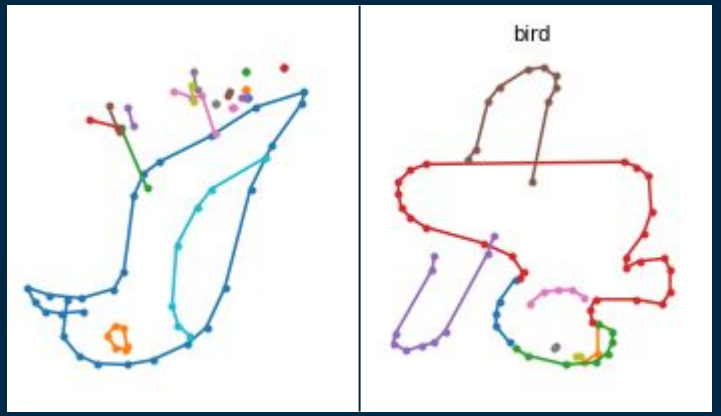


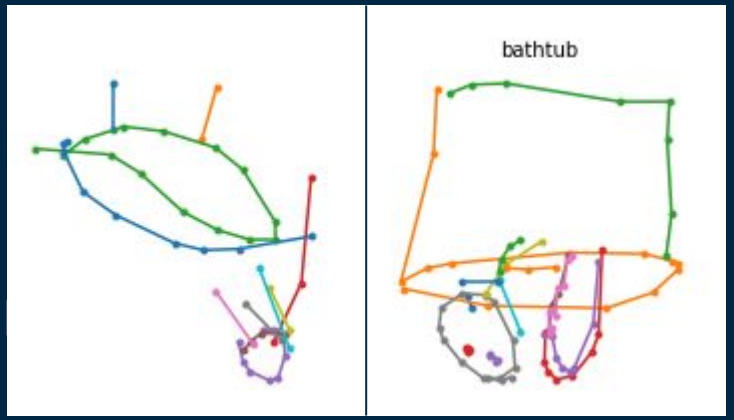


- All drawings are recognized as True
- Red arrows show different direction of drawing
- Form of Data Augmentation
- Helpful in generalizing to new images and reduce variance

More drawing examples (False)





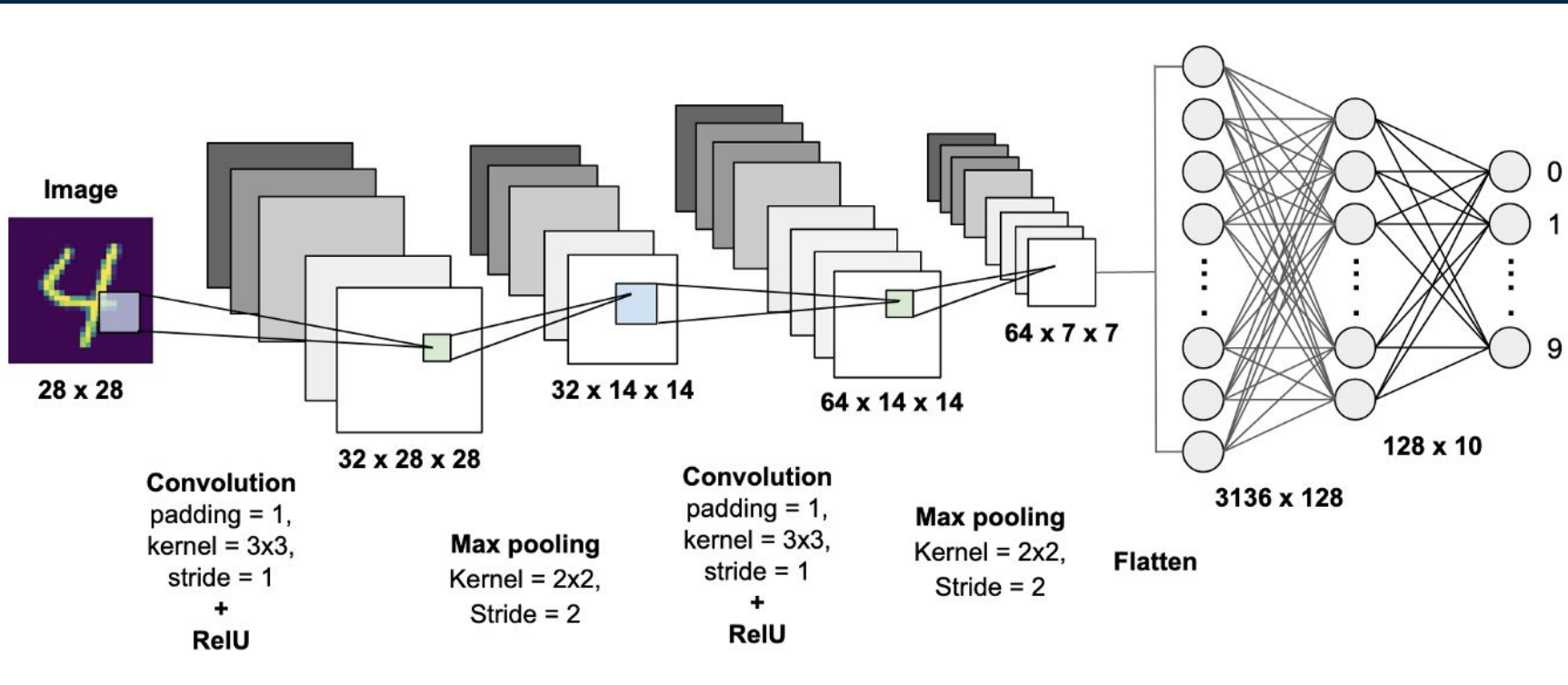


- All drawings are recognized as False
- Most of the drawings do not fit the label
- Exception - bird on the left
- Not including False drawings into the model
- May confuse the model

CNN model & Transfer learning

02

Basic CNN Model Architecture

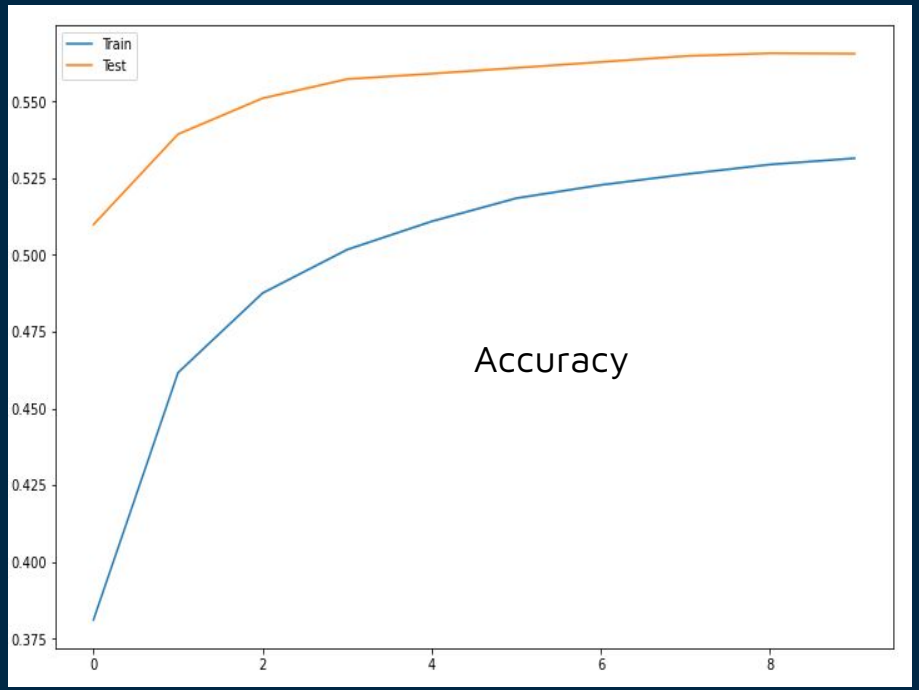


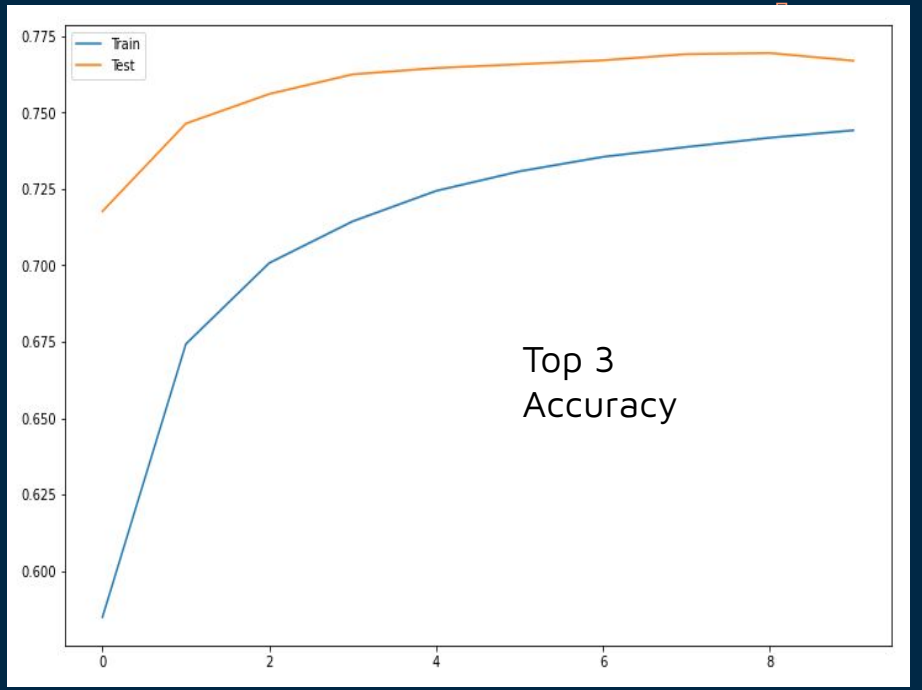
Basic CNN Model Architecture

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 680)               2785960
_________________________________________________________________
dropout (Dropout)            (None, 680)               0
_________________________________________________________________
dense_1 (Dense)              (None, 340)               231540
=================================================================
Total params: 3,036,316
Trainable params: 3,036,316
Non-trainable params: 0
_________________________________________________________________
```

- Image fed in as (32, 32, 1)
- Box in red - giving the 3D image depth and extracting features
- Box in blue - flattens into a 1 dimensional array and gives an output of 340 probabilities
- Not too many layers as training time increases exponentially
- Used 2000 image per class due to limited RAM

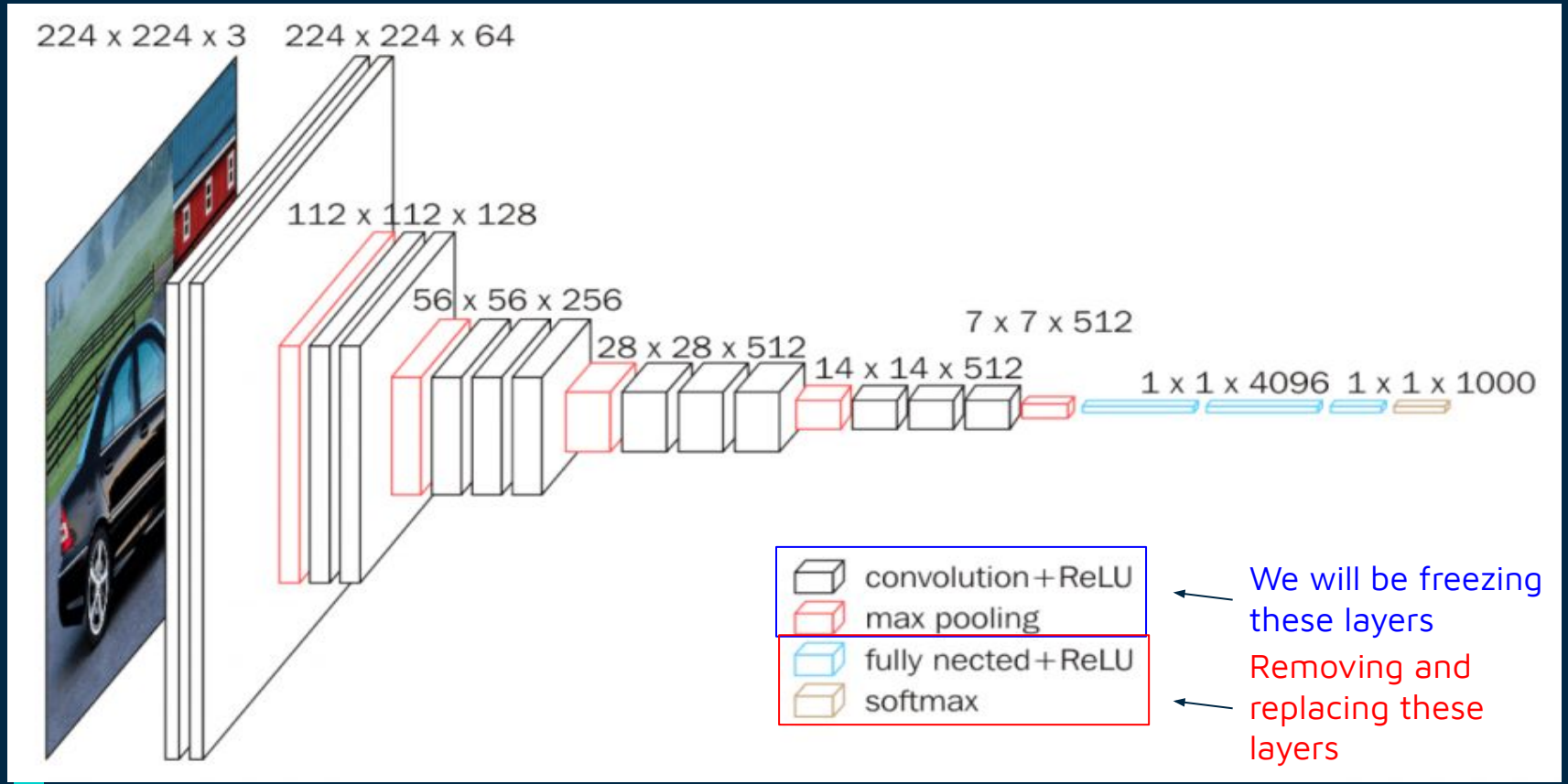
Basic CNN Model Performance





- Validation Accuracy of 0.56, Top 3 Accuracy of 0.76
  - No overfitting - lines slowly converging
  - Test is performing better - low variance

Transfer Learning – VGG16



Transfer Learning – VGG16

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
3layers_for_vgg (Conv2D)     (None, 32, 32, 3)         30
_________________________________________________________________
vgg16 (Functional)           (None, 1, 1, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 680)               348840
_________________________________________________________________
dropout (Dropout)            (None, 680)               0
_________________________________________________________________
dense_1 (Dense)              (None, 340)               231540
=================================================================
Total params: 15,295,098
Trainable params: 580,410
Non-trainable params: 14,714,688
_________________________________________________________________
```

- Image fed in as (32, 32, 1)
- Box in red - Created a Conv2D layer before VGG16 base model to create 'RGB' image
- Box in blue - flattens into a 1 dimensional array and gives an output of 340 probabilities
- Trained the model on Google Colab
- Used 1000 image per class due to limited RAM

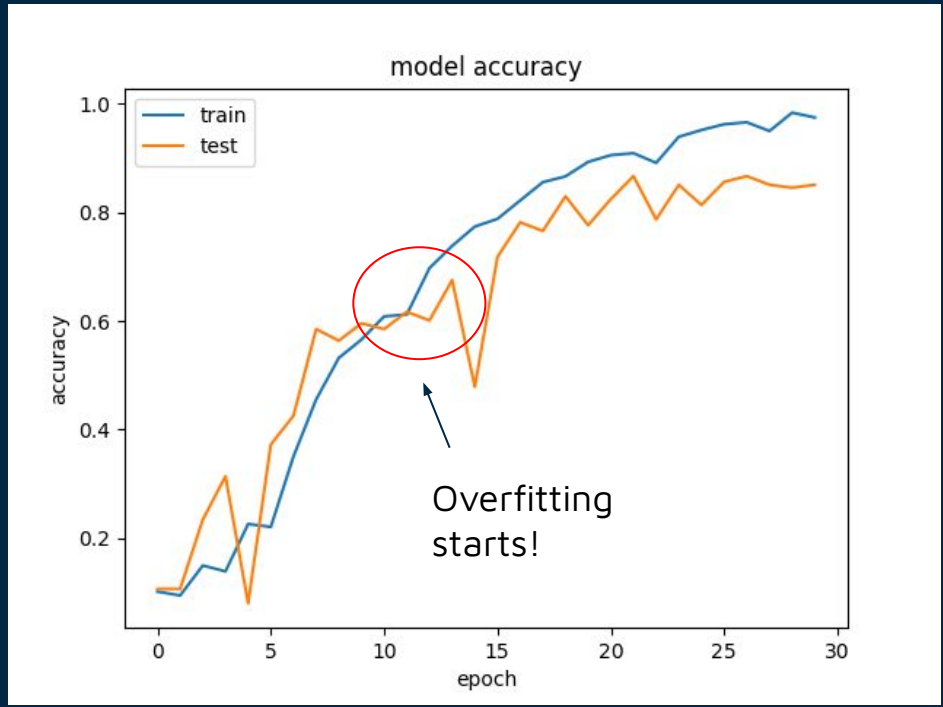
Summary of Model score

Models	Number of image used per class	Train Top 3 Accuracy	Test Top 3 Accuracy	Kaggle Top 3 Accuracy
Basic CNN	2000	0.74	0.76	0.61
VGG16 (32x32)	1000	0.55	0.57	Did not try
VGG16 (128x128)	75 x 5	0.76	**0.78**	**0.62**

- Lesser image used as dimension gets bigger due to limited RAM
- Bigger dimension better score for pre-trained models (VGG16 trained on 224x224x3 images)
- Kaggle score did not perform well - likely due to model not trained on enough images
- Trained on less than 1 million images out of 50 million+
- VGG16 (128x128) model saved, loaded and trained on 5 different sets of 75 images per class

Challenges faced during project

- Basic model run on local CPU took more than 12hrs
- Google colab RAM limit - reduce images in read in
- Pre-trained model overfits very quickly
- Reduce start Learning Rate by a factor of 0.25 to prevent overfitting
- Lastly a lot of rabbit hole



Conclusion & Recommendations

03

Conclusion and Recommendations

- Best model - VGG16 (128x128)
- Model predicts relatively well with 0.62 top 3 accuracy
- Limitations:
  - Only able to predict images from the 340 labels
  - Will predict colored drawings as if it were grayscale
- Further improvements:
  - More time and computing power
  - Read in images as 224x224
  - Train on all 50 million+ images
  - Try out other pre-trained models (MobileNet, EfficientNet)
- Recommendations:
  - Toddler programme/application to guide them in drawing right
  - Learn how to draw and develop artistic talents at an early age

Do you have any questions?

THANK YOU