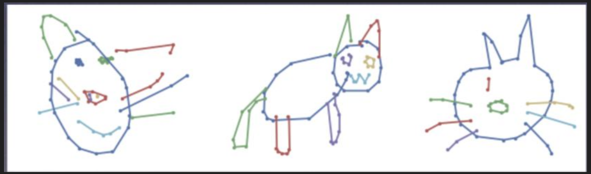
Quick, Draw! Image Classification

Chaitya Shah, Karan Muralidhar, Tyler Brown, Omair Ahmed

Recap on Problem Statement:

- Objective: Google's Quick Draw image classification.
- Dataset: 345 categories, time-series data of strokes, 80GB original dataset, 7GB simplified dataset.
    - Align each drawing to top left corner.
    - Uniformly scale each drawing.
    - Resample all strokes with a 1 pixel spacing.
    - Simplify all strokes using the Ramer-Douglas-Peuker algorithm.
- Preliminary analysis:
    - Data too noisy - drawings of a label can have various correct versions completely unlike each other(Front face of a cat vs side profile of a cat).
    - 
- Proposed possible methods: Neural Networks, CNN

Solution - Data Preprocessing & Strategy

- Data:
    - With original time series data, Google also provides “.npy” files representing the final drawn images
    - Image Resolution: 28*28*1 (784 features)
- Hypothesis:
    - $H1_a$: Fully Connected Neural Network will struggle to get reasonable accuracy.
    - $H2_a$: CNN should perform better than fully connected neural network.
    - $H3_a$: Small amounts of data will lead to overfitting.
    - $H4_a$: Imbalance class prediction can be improved using data augmentation.
    - $H5_a$: Addition of a category to the trained model will reduce the performance of the model.
    - $H6_a$: Adding more training data will improve the performance of the model.

Image Classification using Humans

Human Intelligence Task

- Classified images by hand into 1 of 10 categories
- People: 2
- Time Taken to Classify: ~30 minutes
- Test Accuracy: 87%

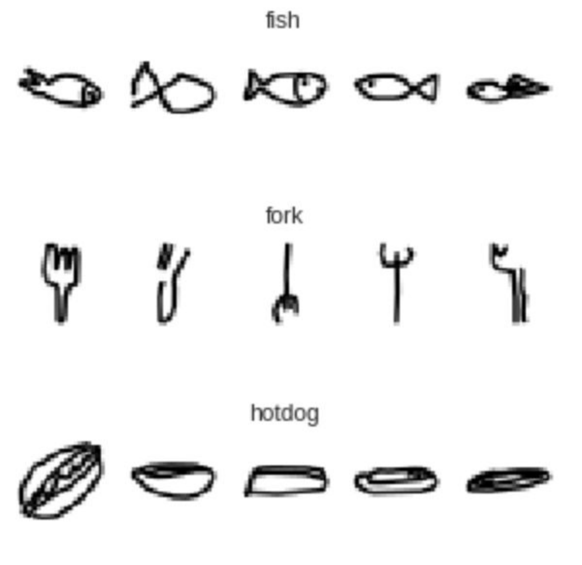


Image Classification using FC-NN

Fully Connected Neural Network Architecture

$$Z_1 = \underset{25\times784}{W_1} X + \underset{25\times1}{b_1} \Rightarrow A_1 = \mathrm{ReLU}(Z_1)$$

$$Z_2 = \underset{12\times25}{W_2} A_1 + \underset{12\times1}{b_2} \Rightarrow A_2 = \mathrm{ReLU}(Z_2)$$

$$Z_3 = \underset{10\times12}{W_3} A_2 + \underset{10\times1}{b_3} \Rightarrow \text{Output Layer} \rightarrow \mathrm{SoftMax}(Z_3)$$

$$(25 \times 784 \underset{Z_1}{+} 25 \times 1) + (12 \times 25 \underset{Z_2}{+} 12 \times 1) + (10 \times 12 \underset{Z_3}{+} 10 \times 1) = 20,067 \text{ parameters}$$

Fully Connected Neural Network Results

- Final result similar slightly less than human accuracy ≅ 87%
  - Training set accuracy ≅86%
  - Test set accuracy ≅ 82%
- Improved overfitting by adding more training data
  - Original training set accuracy ≅ 93%
  - Original test set accuracy ≅ 73%
- Number of epochs: 1,500
- Learning rate α = 0.001

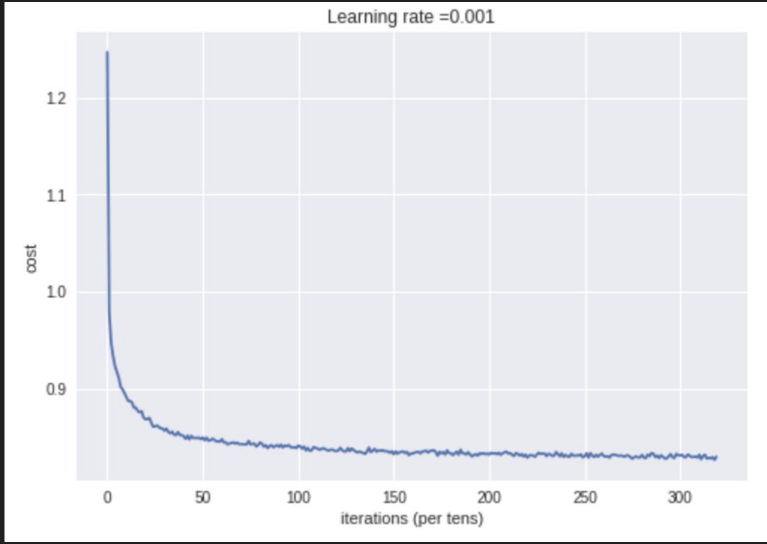


Image Classification using CNN

CNN

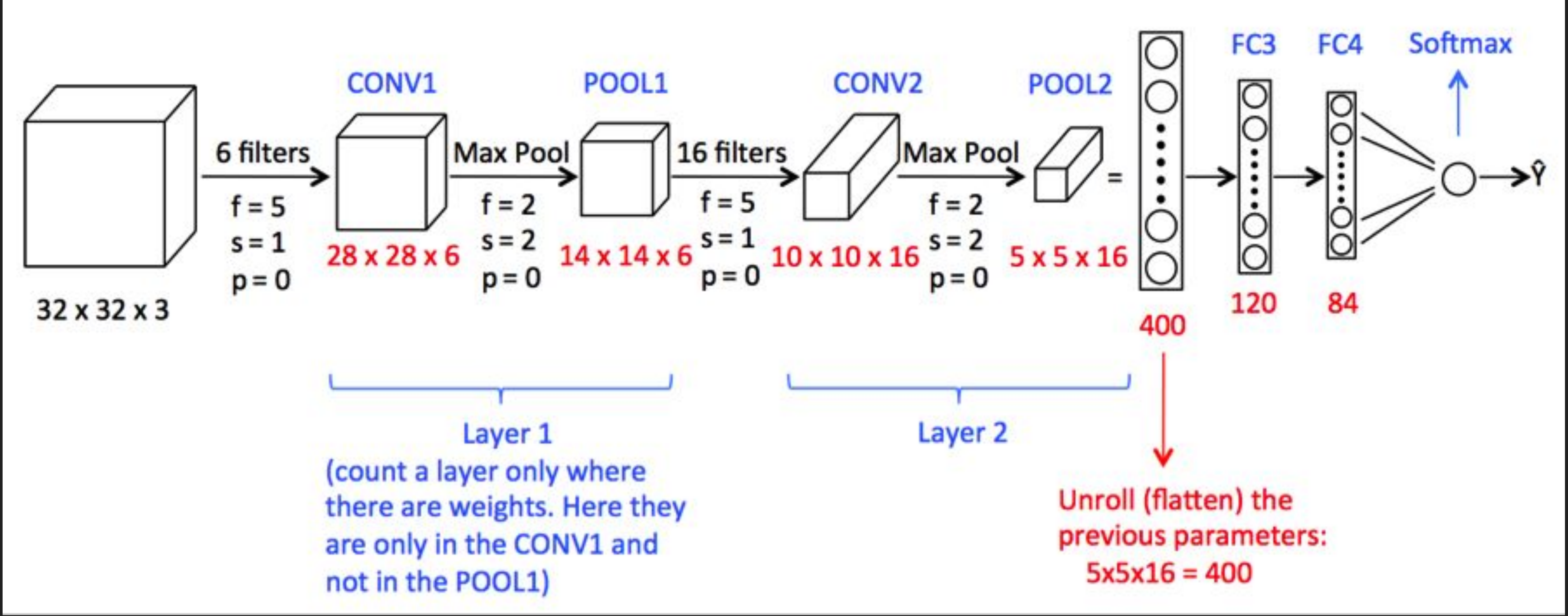
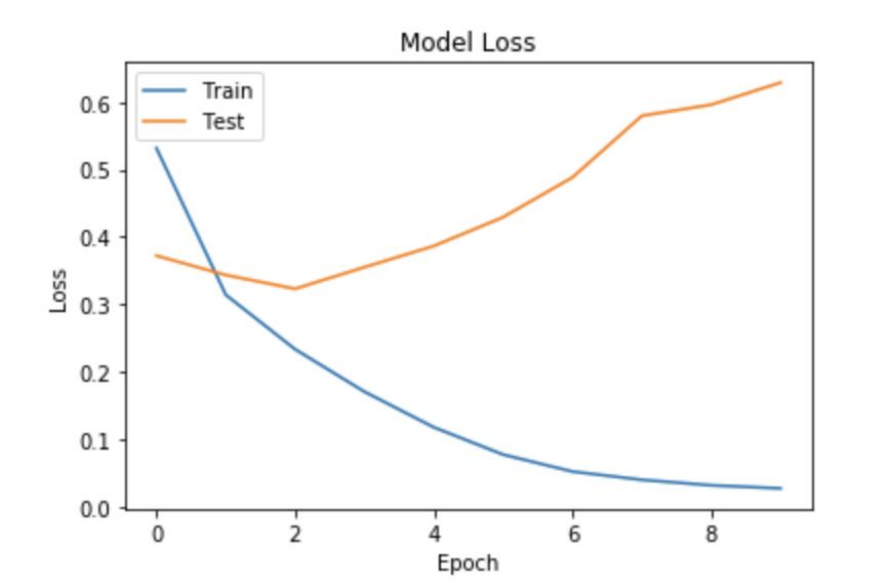
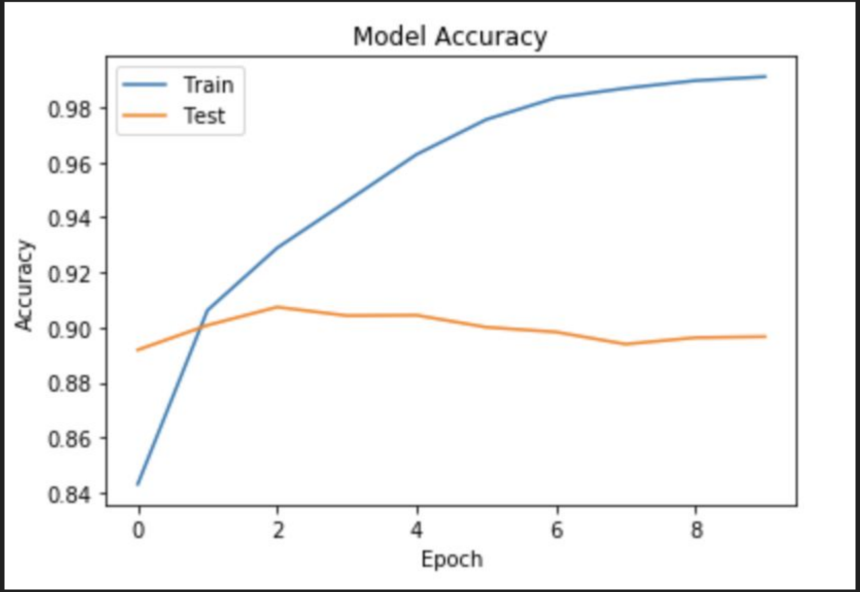


Image obtained from https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524

Model 1: 10 categories, 100,000 observation, 10 epochs

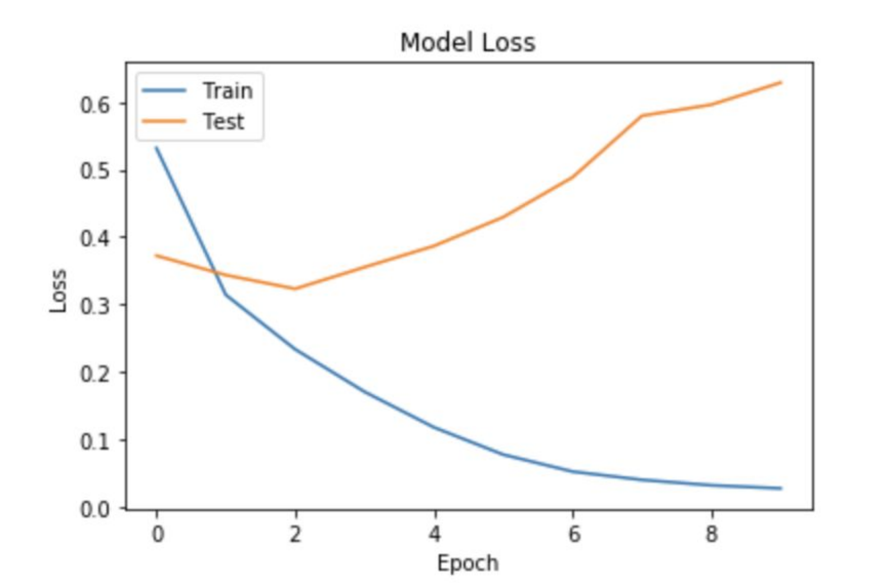
Layer1: Conv2d(3*3*16), Relu, **Layer2**: Conv2d(2*2*32), Relu, **Layer3**: Dense(64), Number of parameters - 1,282,954

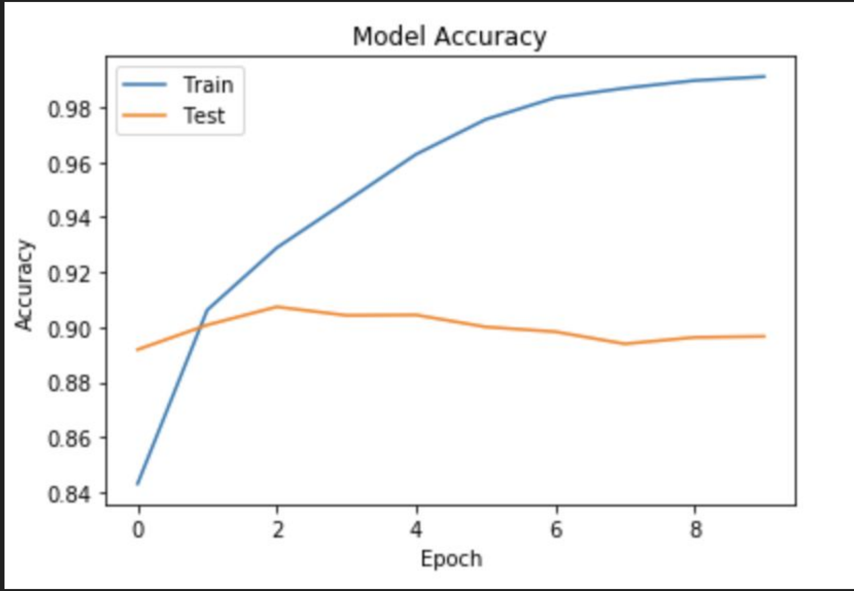




Model 2: 10 categories, 100,000 observation, 10 epochs

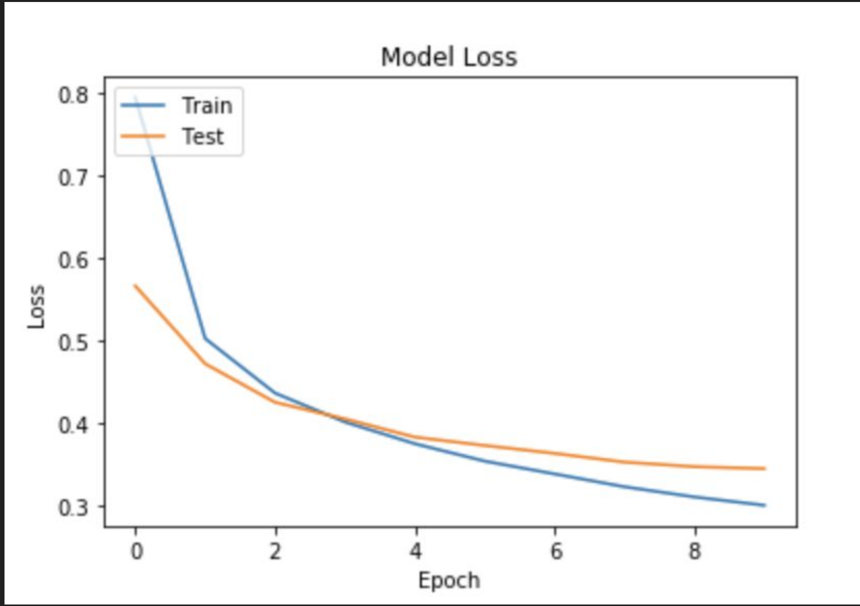
Layer1: Conv2d(3*3*4), Relu, **Layer2**: Conv2d(2*2*8), Relu, **Layer3**: Dense(64), Number of parameters - 320,890

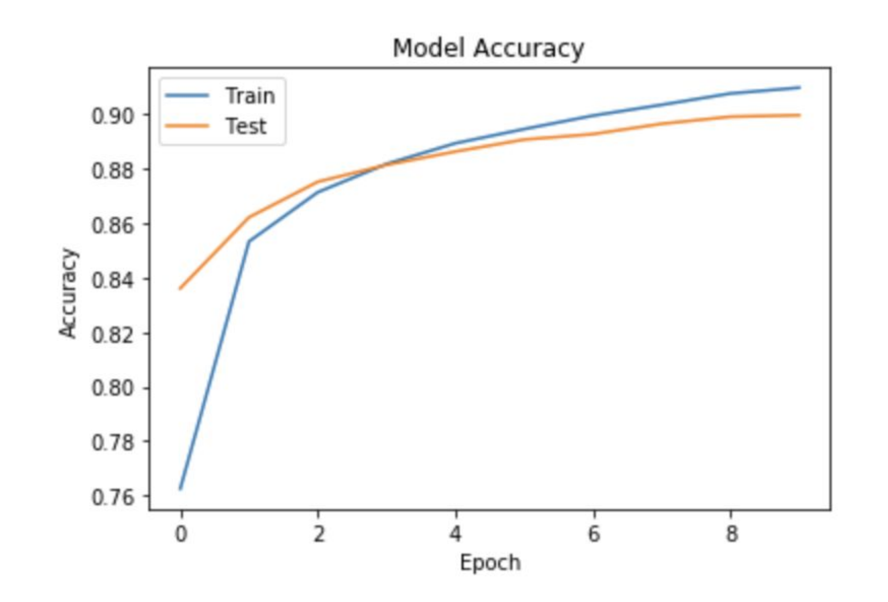




Model 3: 10 categories, 100,000 observation, 10 epochs

Layer1: Conv2d(3*3*4):MaxPool(2,2), Relu, **Layer2**: Conv2d(2*2*8):MaxPool(2,2), Relu, **Layer3**: Dense(64), Number of parameters - 19,322





Model: Padding, Stride, Dropout

- Padding
  - Improvement in accuracy(train, test) at the cost of parameters.
- Stride
  - Drastic decrease in the number of parameters to 2000 because of usage of small number of feature detectors(depth).
- Dropout
  - Takes longer time to converge(40 epochs).

Overfitting

Layer1: Conv2d(3*3*4):MaxPool(2,2), Relu, **Layer2**: Conv2d(2*2*8):MaxPool(2,2), Relu, **Layer3**: Dense(64), Number of parameters - 19,322

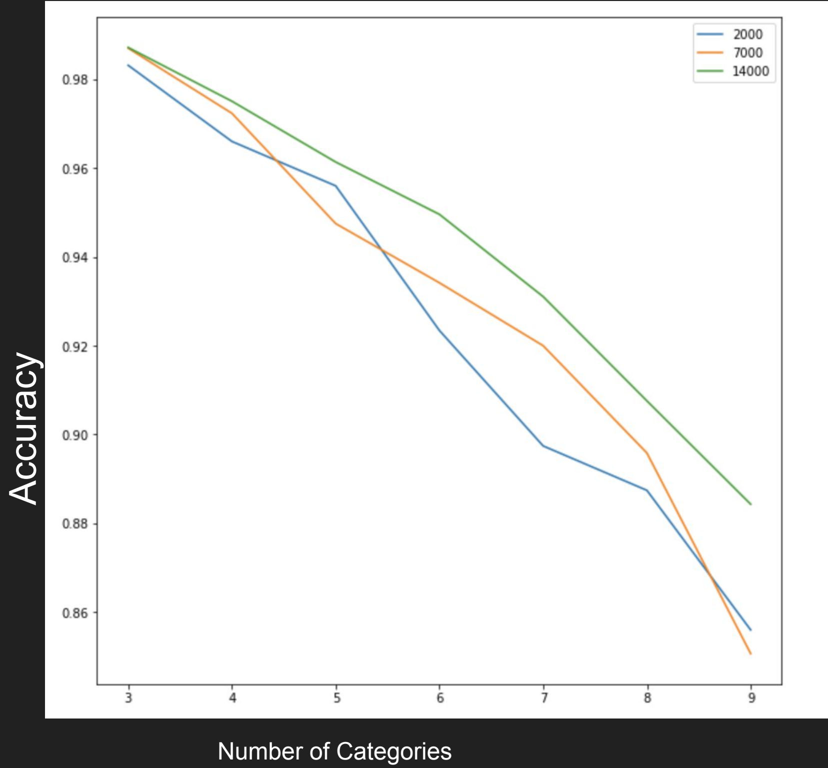
Data(10 categories, epochs-10)	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
500	(0.39, 0.92)	(0.58, 0.82)	(1.93, 0.25)	(1.45, 0.63)
1000	(0.46, 0.91)	(0.62, 0.83)	(1.66, 0.28)	(1.22, 0.63)
3000	(0.66, 0.93)	(0.77, 0.87)	(1.09, 0.22)	(0.77, 0.87)
5000	(0.69, 0.93)	(0.79, 0.89)	(0.99, 0.22)	(0.66, 0.41)
7000	(0.69, 0.93)	(0.80, 0.89)	(0.98, 0.22)	(0.70, 0.38)
10000	(0.76, 0.91)	(0.83, 0.89)	(0.79, 0.20)	(0.56, 0.35)

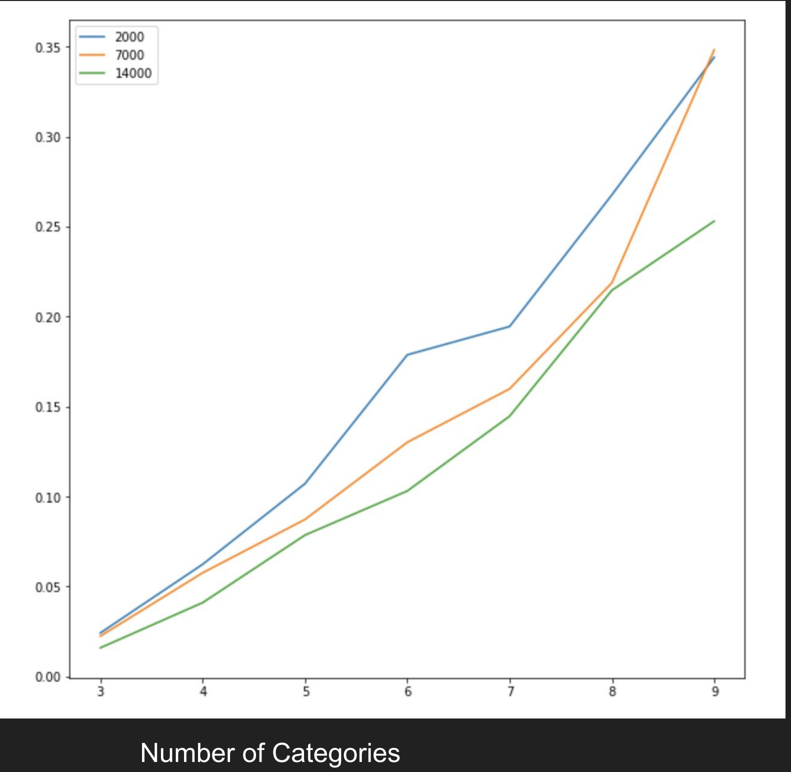
Effect of Number of Categories vs Rows per Category

- Increase the **number of categories** the Neural Network had to model, holding the number of **examples per categories** constant.
- Increase **examples per categories** holding the **number of categories constant** in steps of 1000 from 7000 to 20000 examples.
- Number of categories ranged from 3 to 10 in steps of 1.

Accuracy

Loss





Quantify the Effect on Accuracy/Loss Per Category

- Run a regression on all 78 examples
- Variable 1: Number of rows per category
- Variable 2: Number of Categories
- The parameters will be interpreted as the effect of accuracy by altering the number of categories and number of rows
- The same was calculated for loss

Results: Accuracies

OLS Regression Results

Dep. Variable:	accuracy	**R-squared:**	0.955
Model:	OLS	**Adj. R-squared:**	0.954
Method:	Least Squares	**F-statistic:**	944.6
Date:	Fri, 07 Dec 2018	**Prob (F-statistic):**	3.40e-60
Time:	00:49:33	**Log-Likelihood:**	307.83
No. Observations:	91	**AIC:**	-609.7
Df Residuals:	88	**BIC:**	-602.1
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	1.0329	0.003	308.957	0.000	1.026	1.040
no_categories	-0.0188	0.000	-42.922	0.000	-0.020	-0.018
rows_per_category	1.601e-05	2.34e-06	6.840	0.000	1.14e-05	2.07e-05

Omnibus:	17.147	**Durbin-Watson:**	0.998
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	21.716
Skew:	-0.925	**Prob(JB):**	1.93e-05
Kurtosis:	4.519	**Cond. No.**	3.39e+03

For every additional category, we can expect the test accuracy go down by 1.88%

For every additional datapoint, we can expect the accuracy to go up by .0016%

The results are statistically significant

Results: Loss

For every additional category, we can expect the test loss go up by 6.4%

For every additional datapoint, we can expect the loss to go down by .0056%

The results are statistically significant

OLS Regression Results

Dep. Variable:	accuracy	**R-squared:**	0.953
Model:	OLS	**Adj. R-squared:**	0.952
Method:	Least Squares	**F-statistic:**	885.3
Date:	Fri, 07 Dec 2018	**Prob (F-statistic):**	5.16e-59
Time:	00:50:15	**Log-Likelihood:**	192.08
No. Observations:	91	**AIC:**	-378.2
Df Residuals:	88	**BIC:**	-370.6
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1101	0.012	-9.231	0.000	-0.134	-0.086
no_categories	0.0649	0.002	41.536	0.000	0.062	0.068
rows_per_category	-5.628e-05	8.35e-06	-6.740	0.000	-7.29e-05	-3.97e-05

Omnibus:	22.158	**Durbin-Watson:**	0.938
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	31.870
Skew:	1.092	**Prob(JB):**	1.20e-07
Kurtosis:	4.906	**Cond. No.**	3.39e+03

Imbalanced Data

- Trained the model using different imbalanced distributions of 2 classes.
- Results of 100,000 samples of ‘fork’ and different number of samples for ‘hammer’ are as below.
- Training and testing accuracies are for the data used to fit the model and the last 2 columns show accuracy on completely unseen data (20000 samples).

Fork samples	Hammer samples	Train accuracy (%)	Test accuracy (%)	Fork accuracy (%)	Hammer Accuracy (%)
100000	1000	99	99	99	63
100000	5000	99	99	99	85
100000	25000	98	98	99	90
100000	50000	98	98	99	94
100000	75000	98	97	97	97
100000	100000	98	97	98	98

Imbalanced Data

- Similarly, here are our results from iterations involving 50,000 samples of ‘fork’ and different number of samples for ‘hammer’.
- Result- As expected, having imbalanced data results in inaccurate results for the undersampled class for unseen data.
- The prediction accuracy isn’t too bad as neural networks can generalize well on the most important weights.

Fork samples	Hammer samples	Train accuracy (%)	Test accuracy (%)	Fork accuracy (%)	Hammer Accuracy (%)
50000	100	99	99	100	43
50000	1000	99	99	99	81
50000	5000	99	98	99	88
50000	10000	98	98	98	94
50000	25000	98	97	98	95
50000	50000	98	97	98	98

Results

Neural networks, in particular convolutional, are good for classifying images.

- Fully connected Neural Network needed a lot more time to give an accuracy comparable to CNN. (1600 epochs vs 10 epochs).
- CNN classification > human classification > fully connected classification
- Training model with less amount of data results in overfitting.
- Data augmentation helped in solving the problem of imbalanced data.

Future Work

- Training the model for larger number of categories and data.
- Make use of time-series data to predict objects while they are drawn.

Thank You