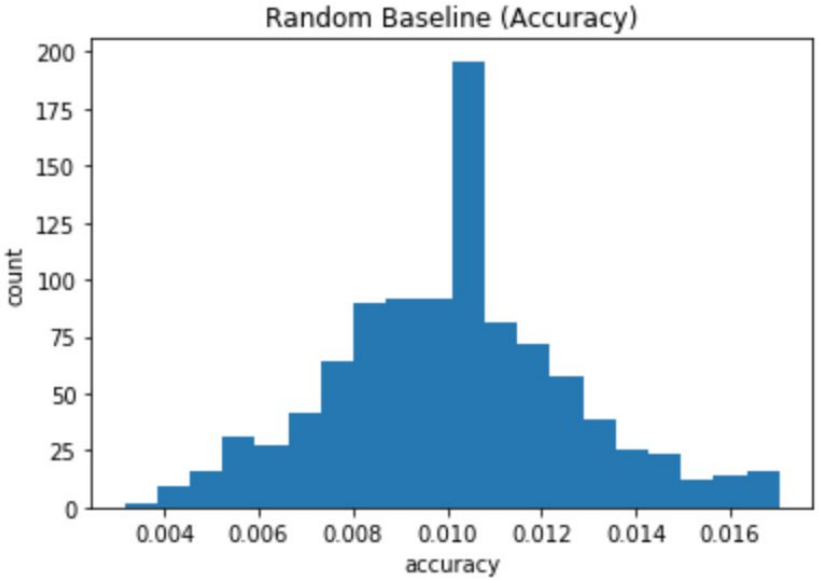
Landmark recognition

Introduction

We’ll train a landmark recognition model using the Google Landmark Dataset containing 4,132,914 images belonging to 203,094 classes. We scale the data down to ~81,300 images belonging to 100 classes. And since the image is our only feature we’ll skip the feature selection part.

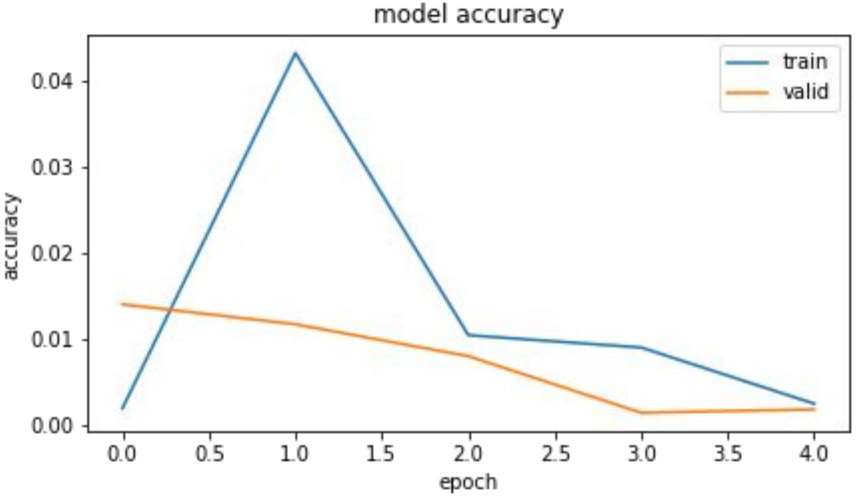
Random Baseline(100 classes)

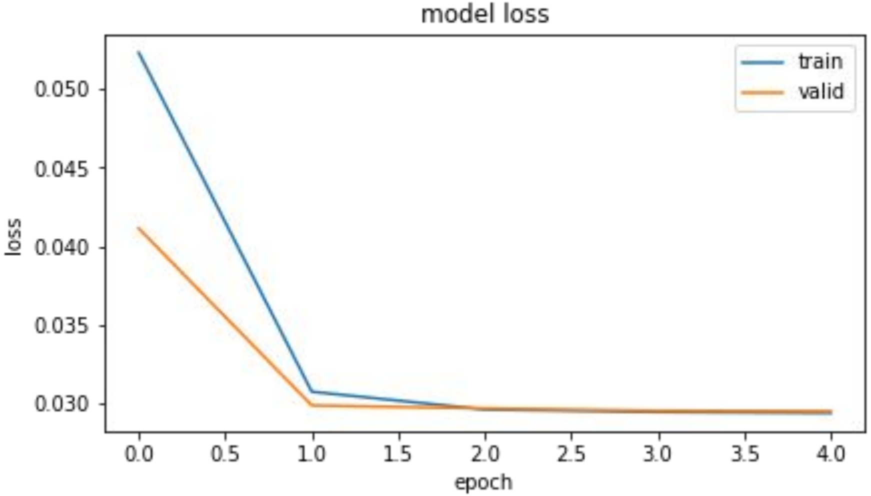


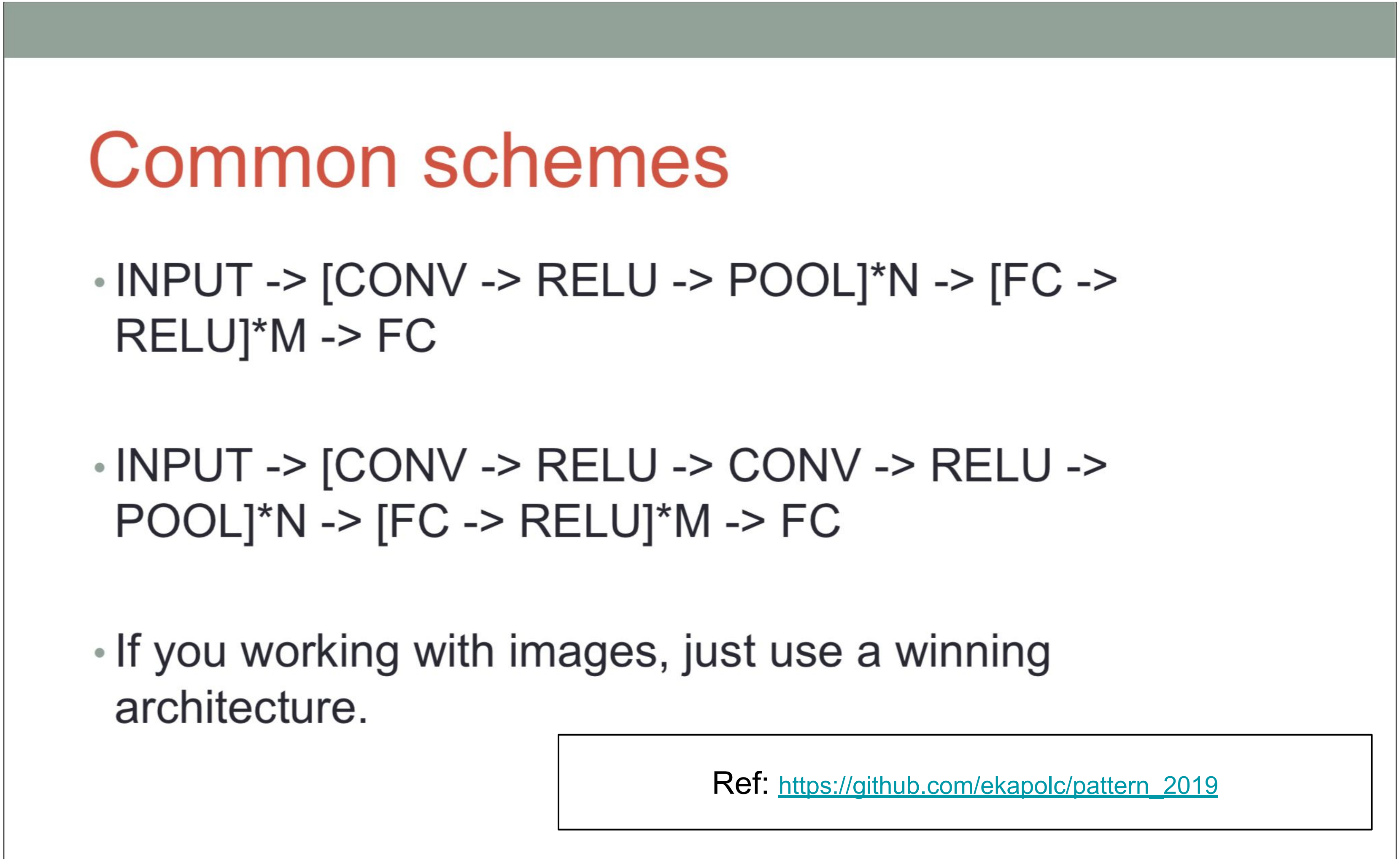
Average Accuracy : 1.01%

Three-Layer Feedforward Neural Networks

Accuracy on validation : 0.1 <%





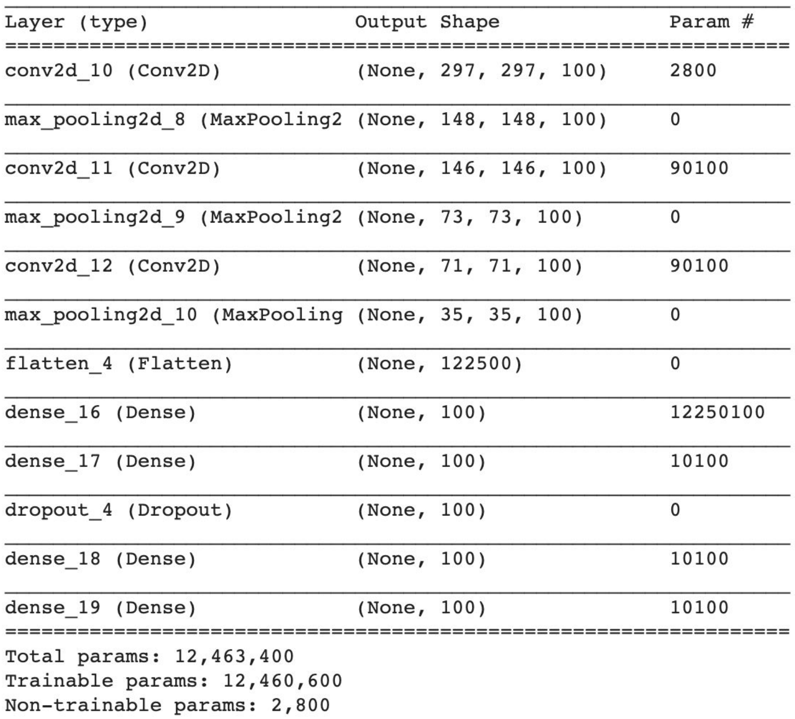


Common schemes

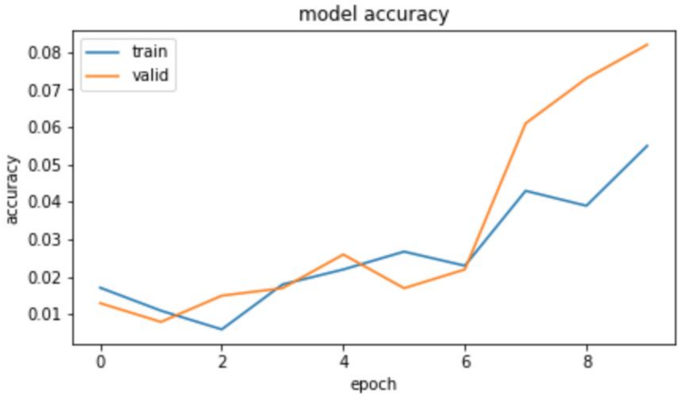
- INPUT -> [CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC
- INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL]*N -> [FC -> RELU]*M -> FC
- If you working with images, just use a winning architecture.

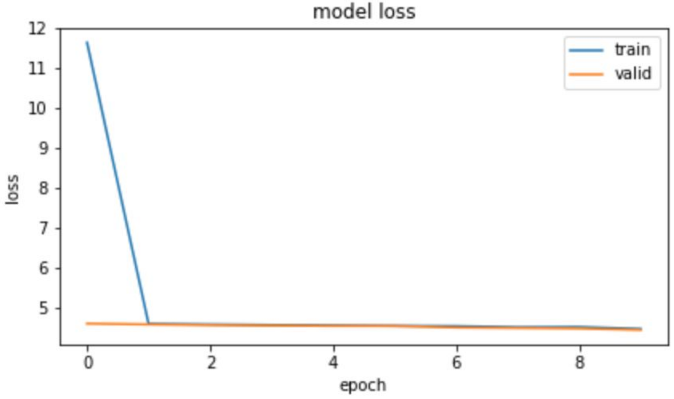
Ref: https://github.com/ekapolc/pattern_2019

CNN with Maxpool

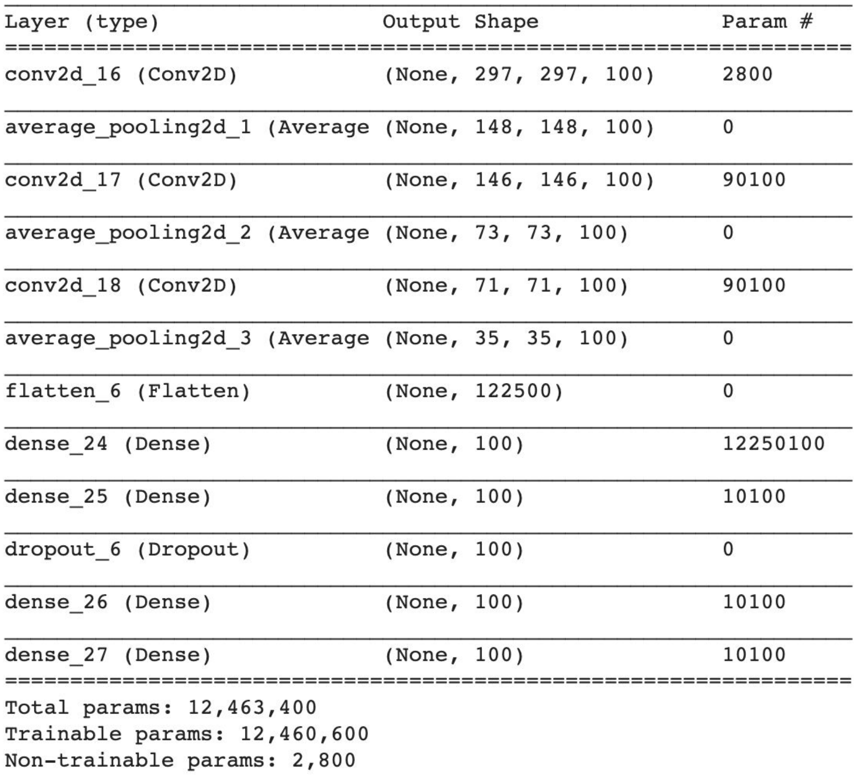


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_10 (Conv2D)           (None, 297, 297, 100)     2800      
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 148, 148, 100)     0         
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 146, 146, 100)     90100     
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 73, 73, 100)       0         
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 71, 71, 100)       90100     
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 35, 35, 100)       0         
_________________________________________________________________
flatten_4 (Flatten)          (None, 122500)            0         
_________________________________________________________________
dense_16 (Dense)             (None, 100)               12250100  
_________________________________________________________________
dense_17 (Dense)             (None, 100)               10100     
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0         
_________________________________________________________________
dense_18 (Dense)             (None, 100)               10100     
_________________________________________________________________
dense_19 (Dense)             (None, 100)               10100     
=================================================================
Total params: 12,463,400
Trainable params: 12,460,600
Non-trainable params: 2,800
```

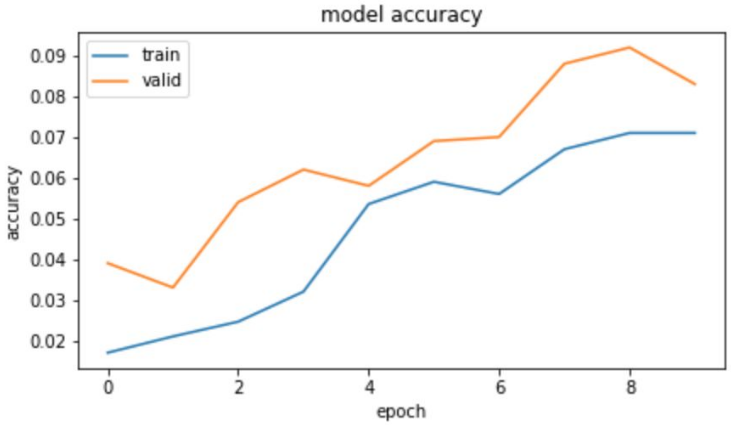


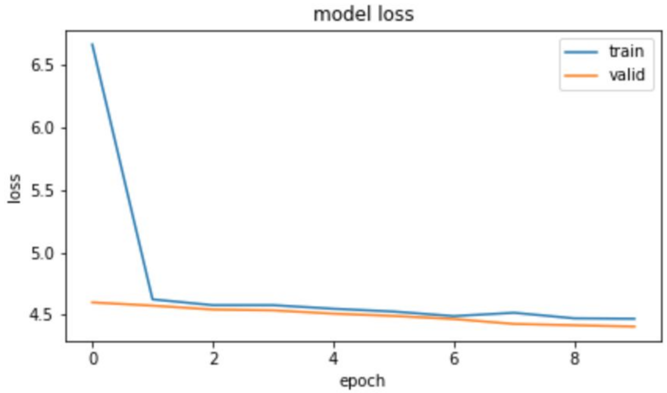


CNN with Averagepool



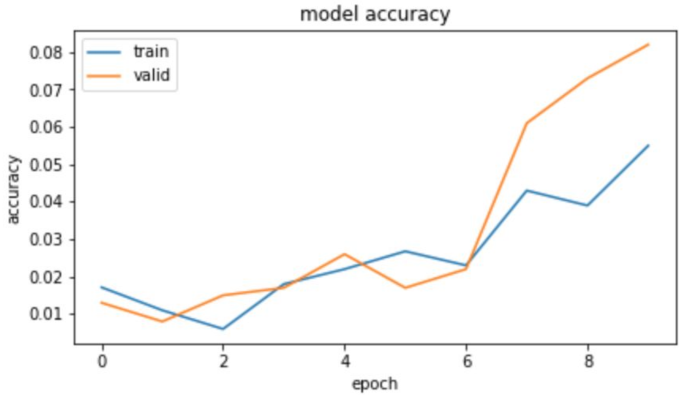
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 297, 297, 100)     2800
_________________________________________________________________
average_pooling2d_1 (Average (None, 148, 148, 100)     0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 146, 146, 100)     90100
_________________________________________________________________
average_pooling2d_2 (Average (None, 73, 73, 100)       0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 71, 71, 100)       90100
_________________________________________________________________
average_pooling2d_3 (Average (None, 35, 35, 100)       0
_________________________________________________________________
flatten_6 (Flatten)          (None, 122500)            0
_________________________________________________________________
dense_24 (Dense)             (None, 100)               12250100
_________________________________________________________________
dense_25 (Dense)             (None, 100)               10100
_________________________________________________________________
dropout_6 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_26 (Dense)             (None, 100)               10100
_________________________________________________________________
dense_27 (Dense)             (None, 100)               10100
=================================================================
Total params: 12,463,400
Trainable params: 12,460,600
Non-trainable params: 2,800
```



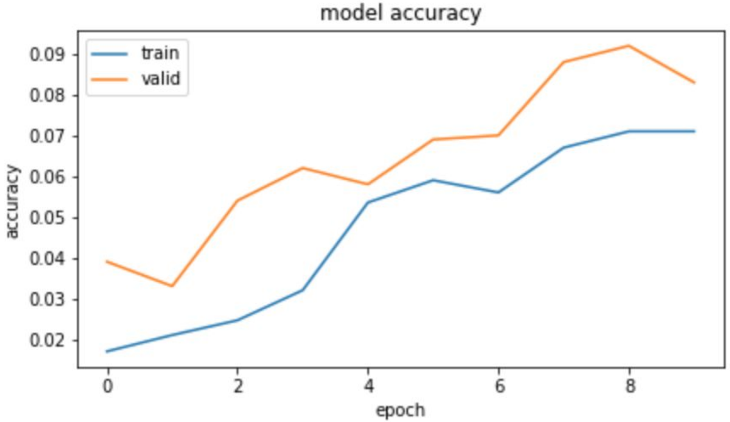


Max VS Average pooling

Max pooling

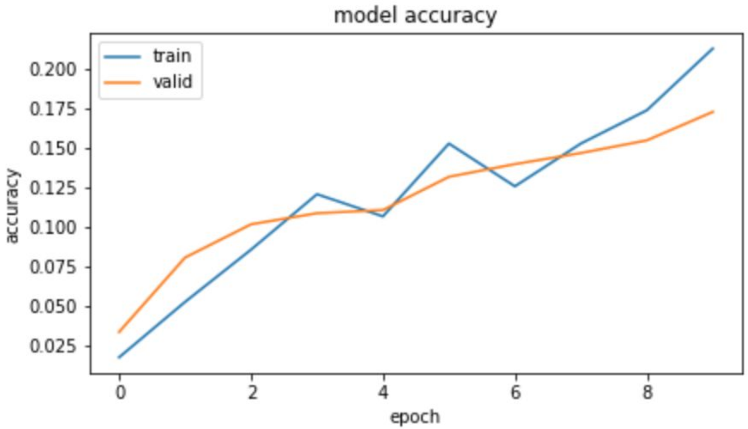


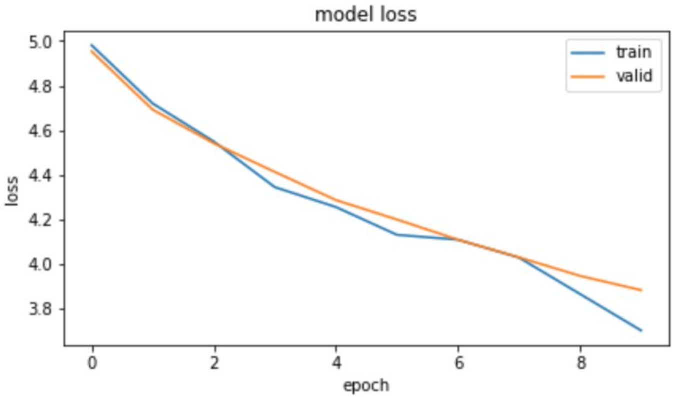
Average pooling



ResNet50 with large dataset

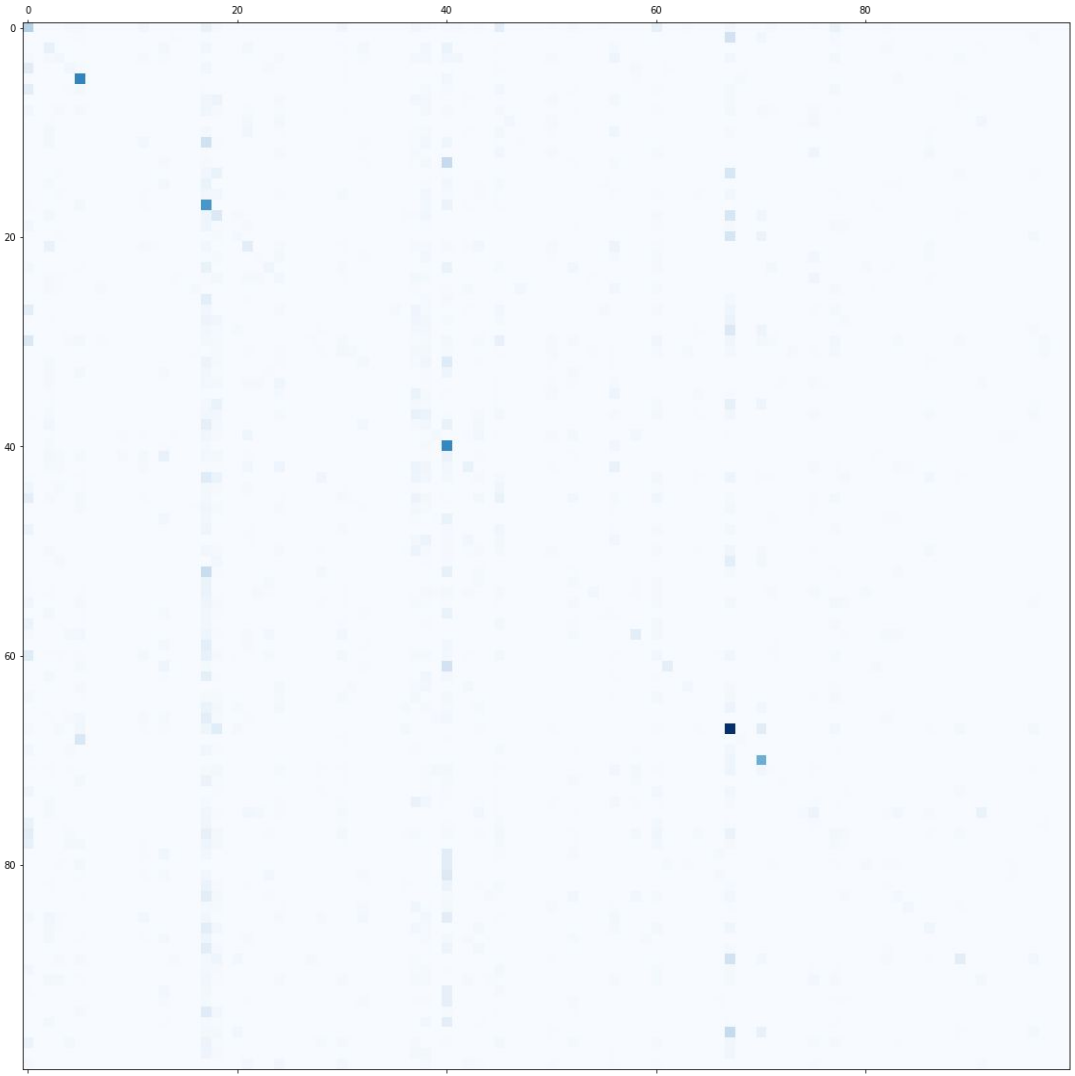
Accuracy on validation : 16.8%





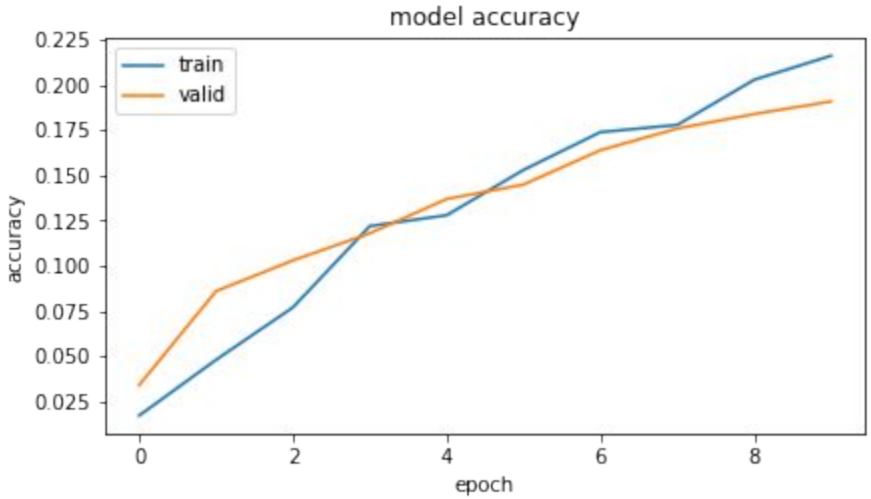
ResNet50 with large dataset

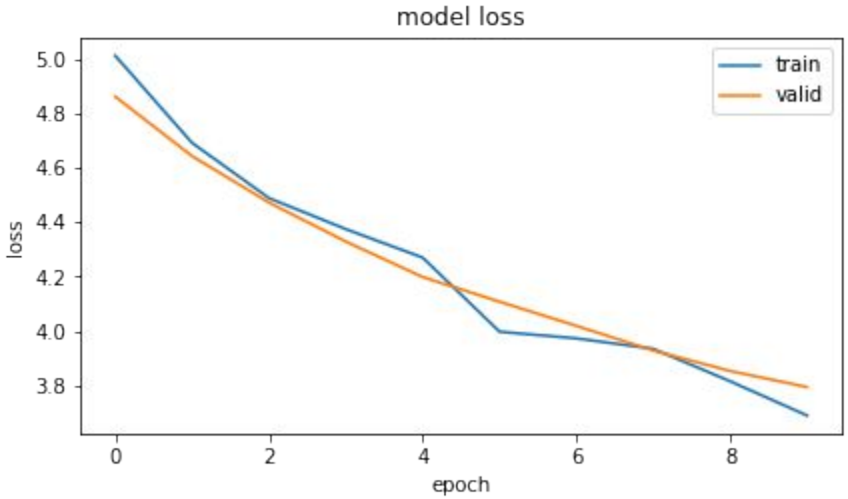
Confusion Matrix



ResNet50 with small dataset

Accuracy on validation : 17.8%





Evaluate on test set(ResNet50 on small dataset)

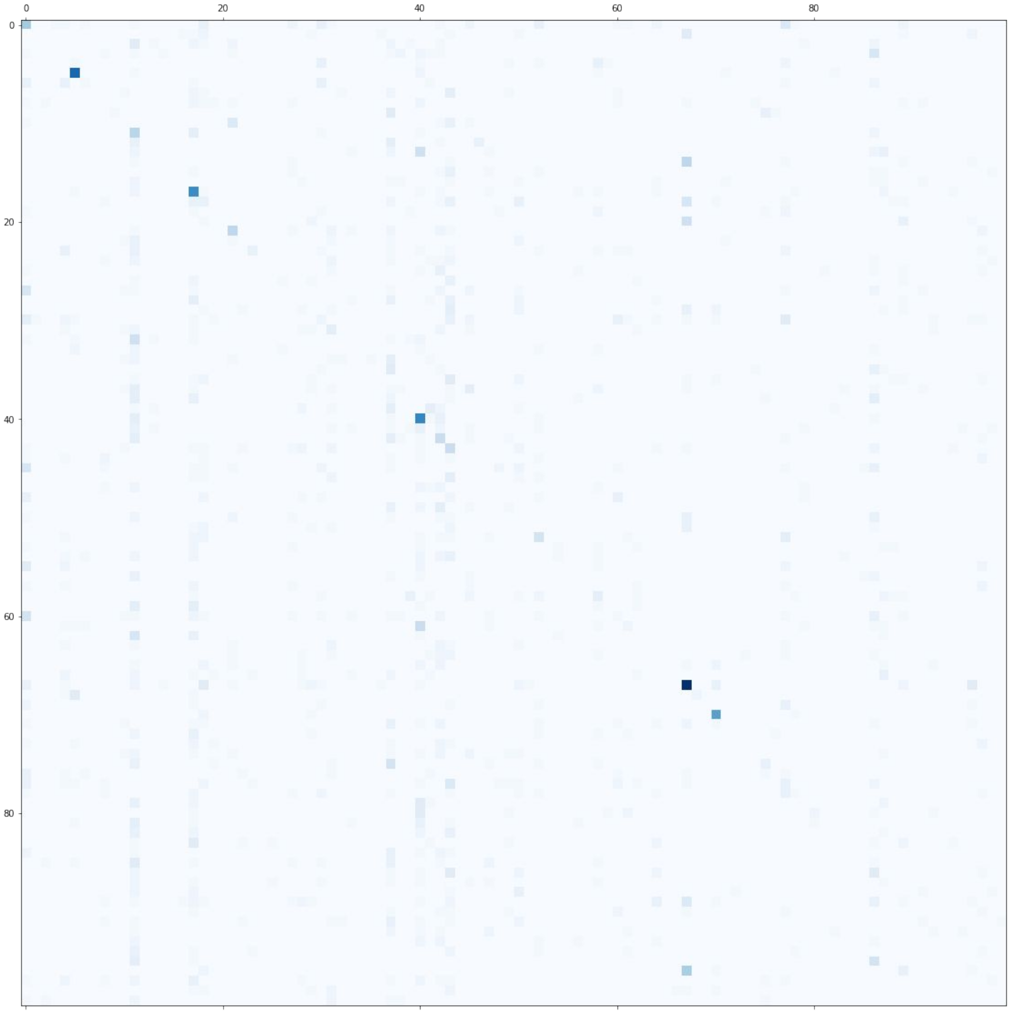
Accuracy top 1: 16.34%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 33.45%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 45.40%
Accuracy majority top 10 : 23.73%

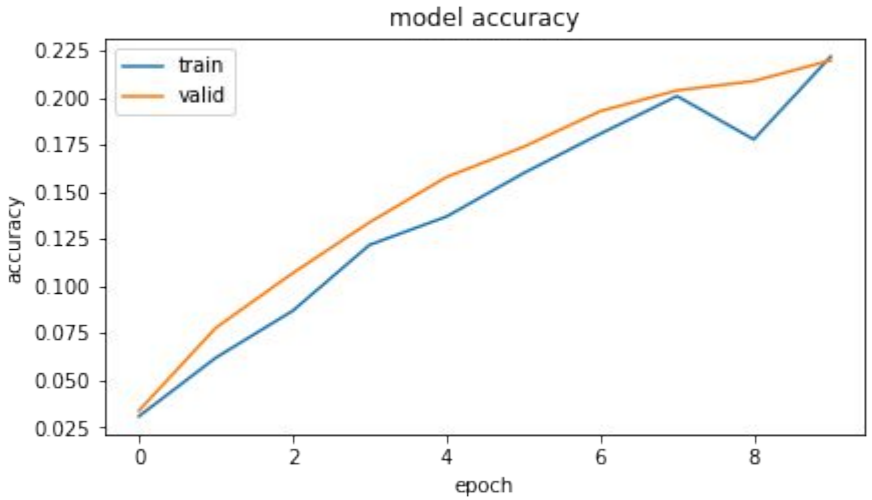
ResNet50 with small dataset

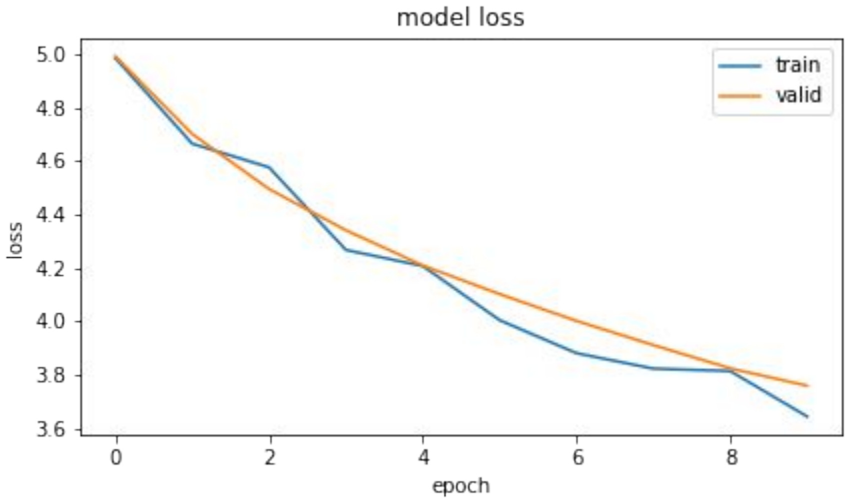
Confusion Matrix



ResNet101

Accuracy on validation : 22.0%





Evaluate on test set(ResNet101)

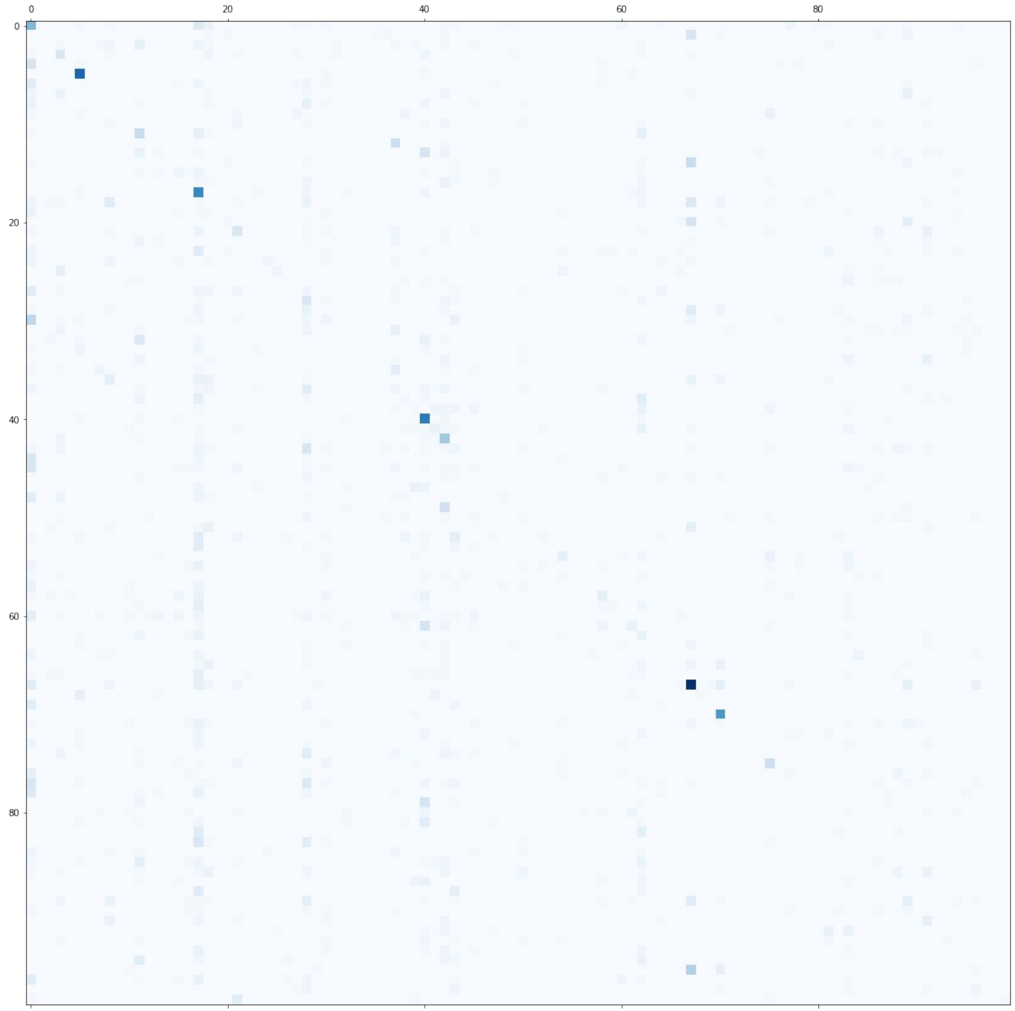
Accuracy top 1: 17.98%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 33.68%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 46.16%
Accuracy majority top 10 : 23.73%

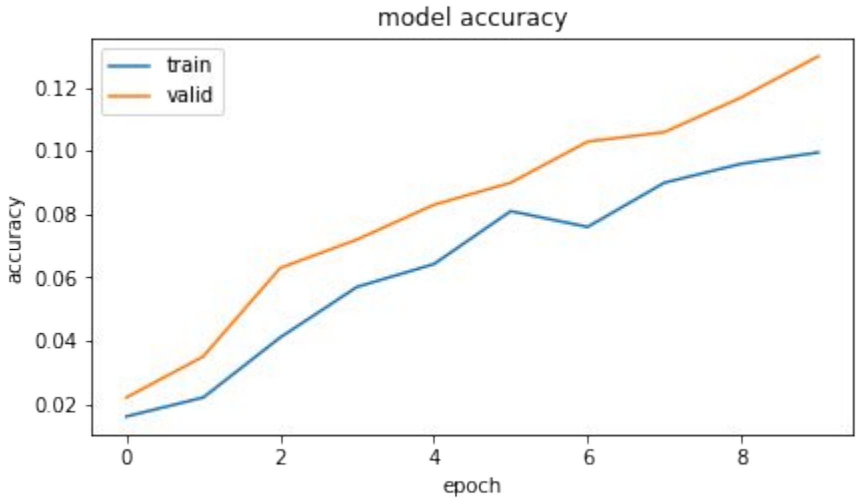
ResNet101

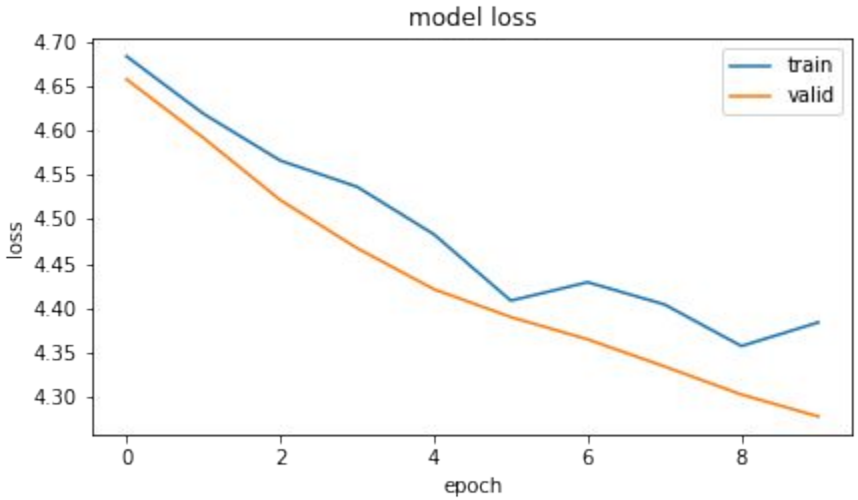
Confusion Matrix



InceptionV3 with sgd

Accuracy on validation : 13.0%





Evaluate on test set(InceptionV3 with sgd)

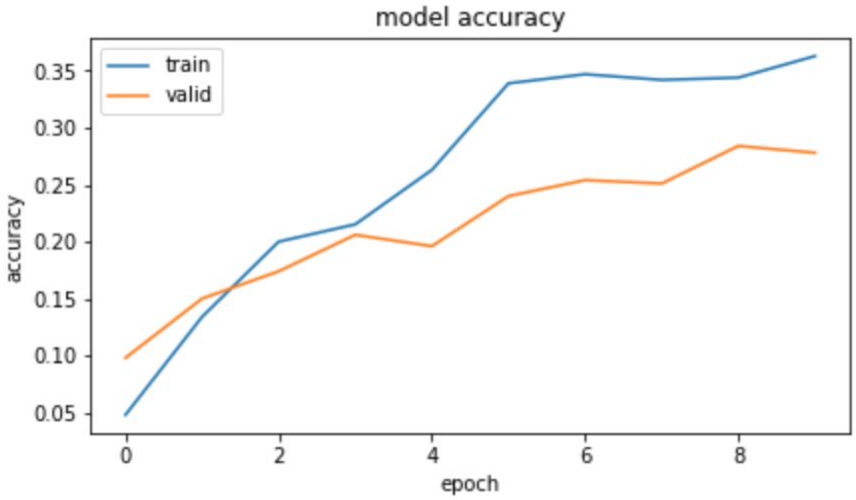
Accuracy top 1: 10.60%
Accuracy majority top 1 : 4.73%

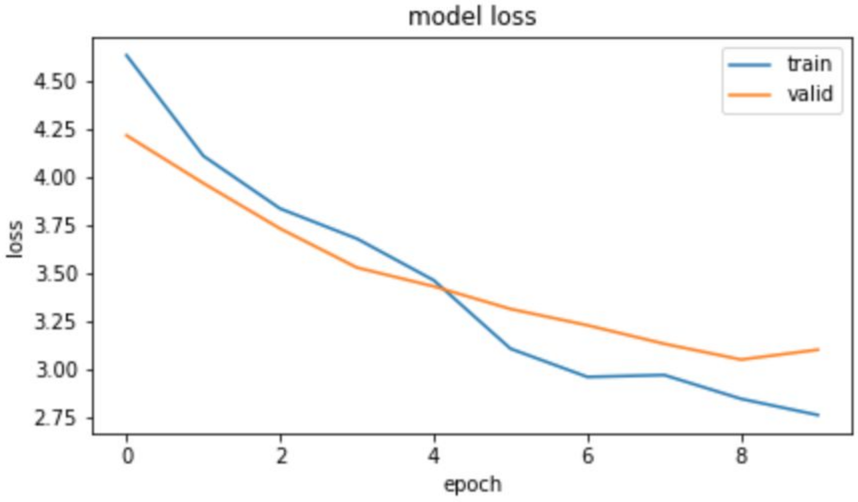
Accuracy top 5 : 20.74%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 29.53%
Accuracy majority top 10 : 23.73%

InceptionV3 with rmsprop

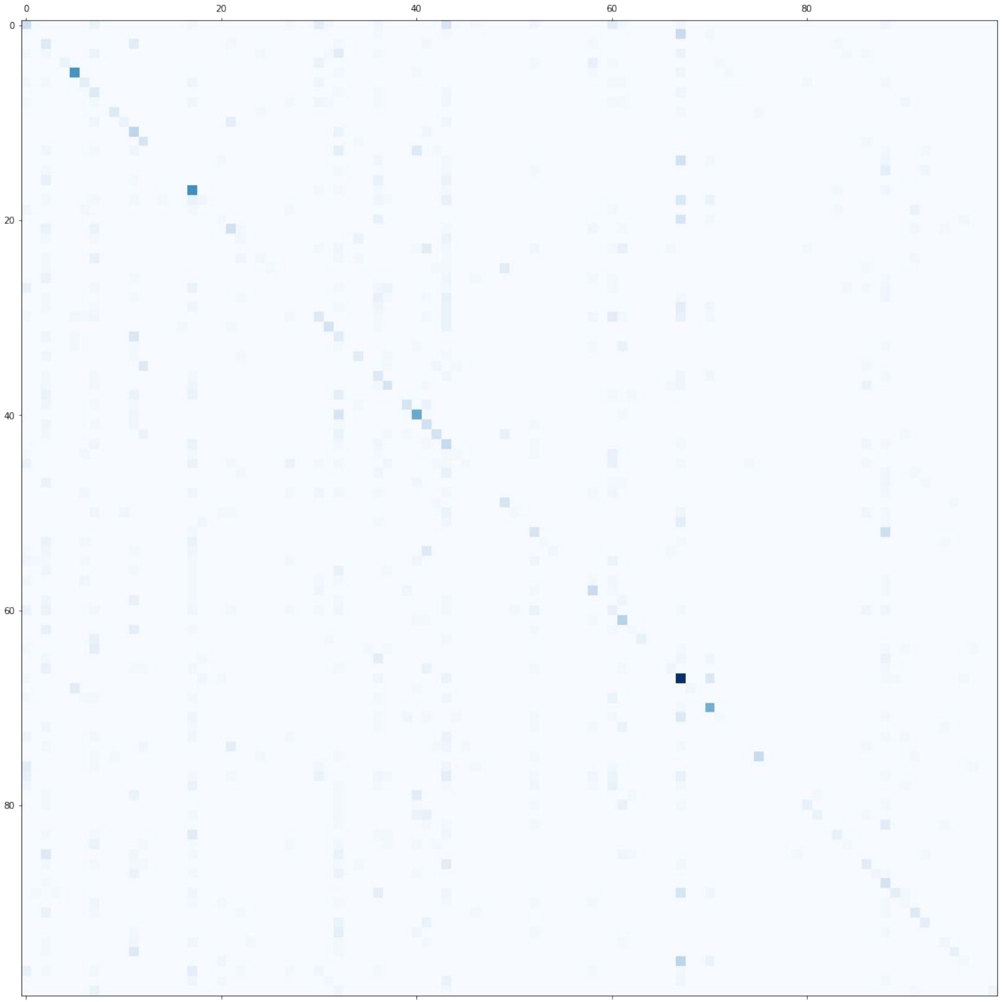
Accuracy on validation : 27.8%





InceptionV3 with rmsprop

Confusion Matrix



Evaluate on test set(InceptionV3 with rmsprop)

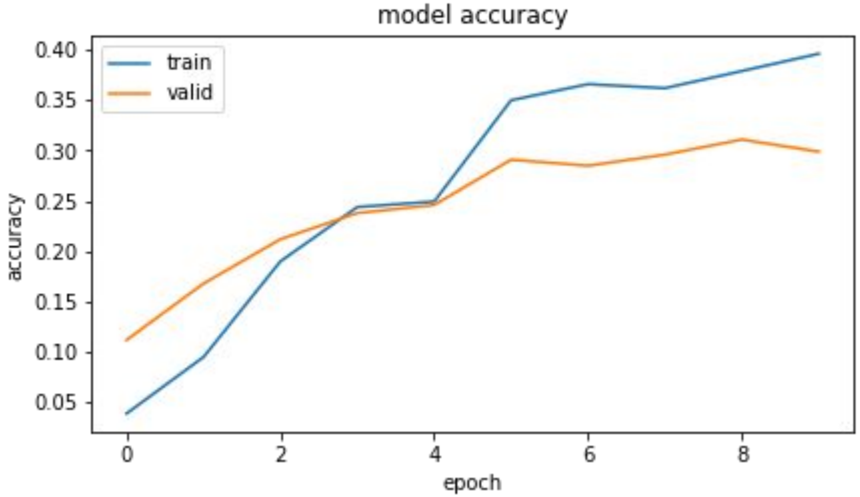
Accuracy top 1: 29.23%
Accuracy majority top 1 : 4.73%

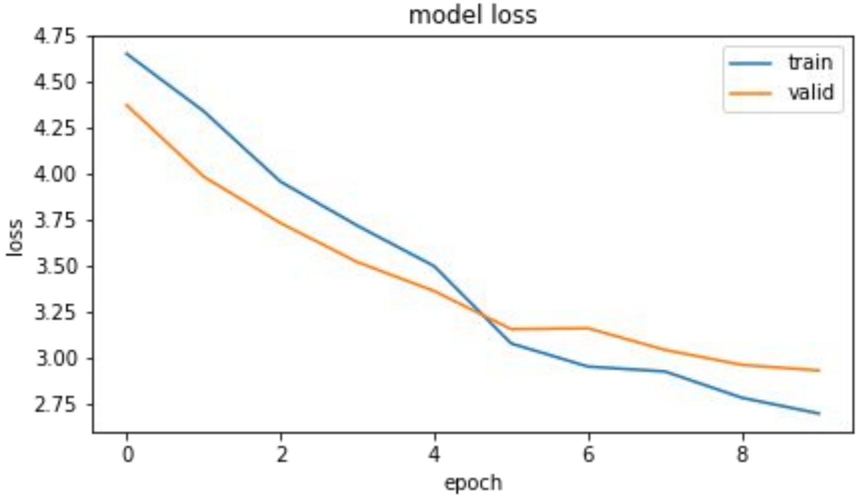
Accuracy top 5 : 54.48%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 69.83%
Accuracy majority top 10 : 23.73%

InceptionV3 with adam

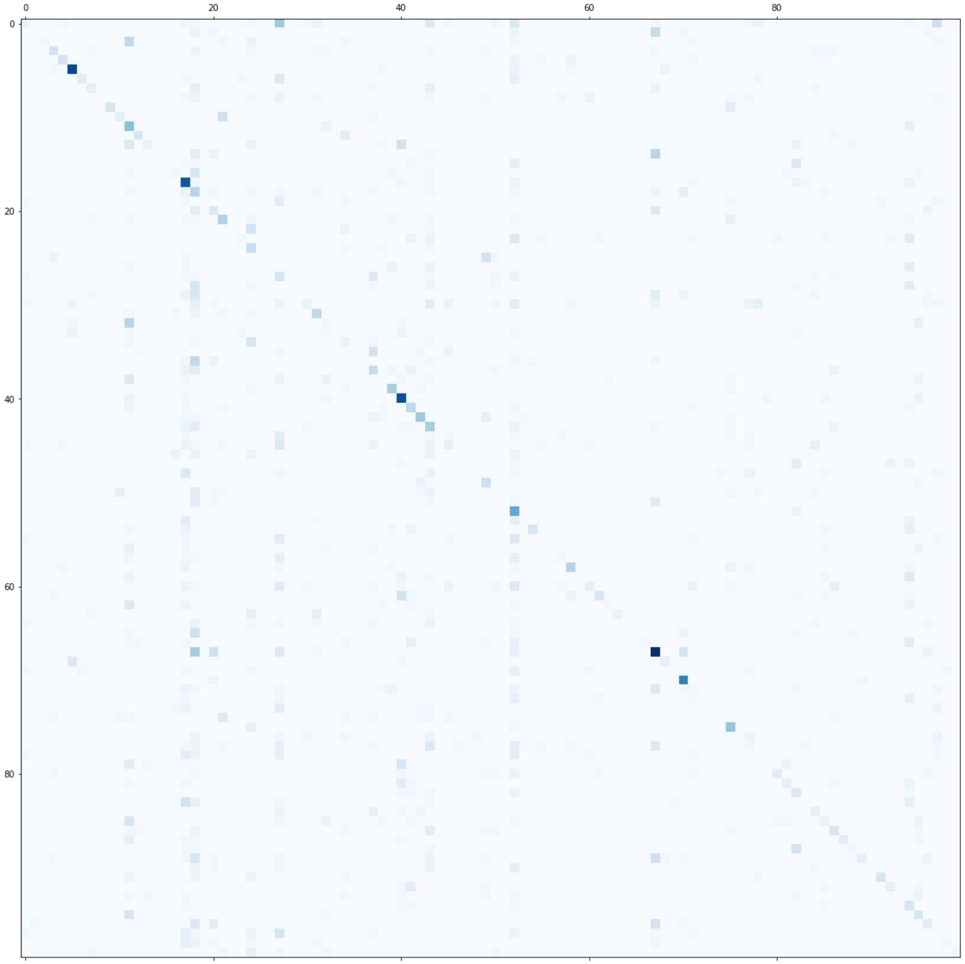
Accuracy on validation : 29.9%





InceptionV3 with adam

Confusion Matrix



Evaluate on test set(InceptionV3 with adam)

Accuracy top 1: 28.06%
Accuracy majority top 1 : 4.73%

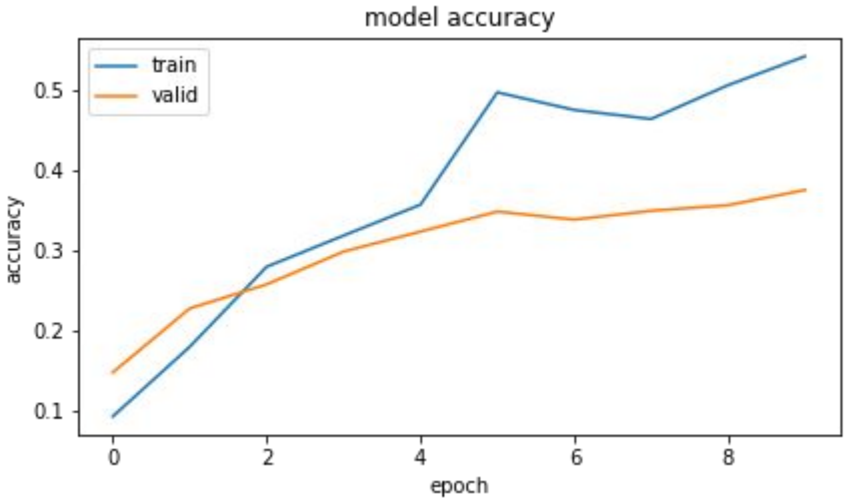
Accuracy top 5 : 57.23%
Accuracy majority top 5 : 15.10%

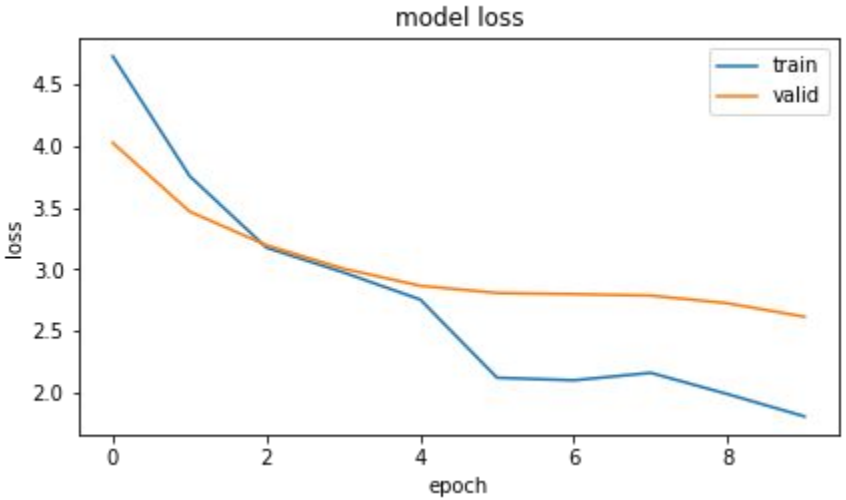
Accuracy top 10 : 69.83%
Accuracy majority top 10 : 23.73%

→ Equal to InceptionV3 with rmsprop

ResNet101 with rmsprop

Accuracy on validation : 37.6%





Evaluate on test set(ResNet101 with rmsprop)

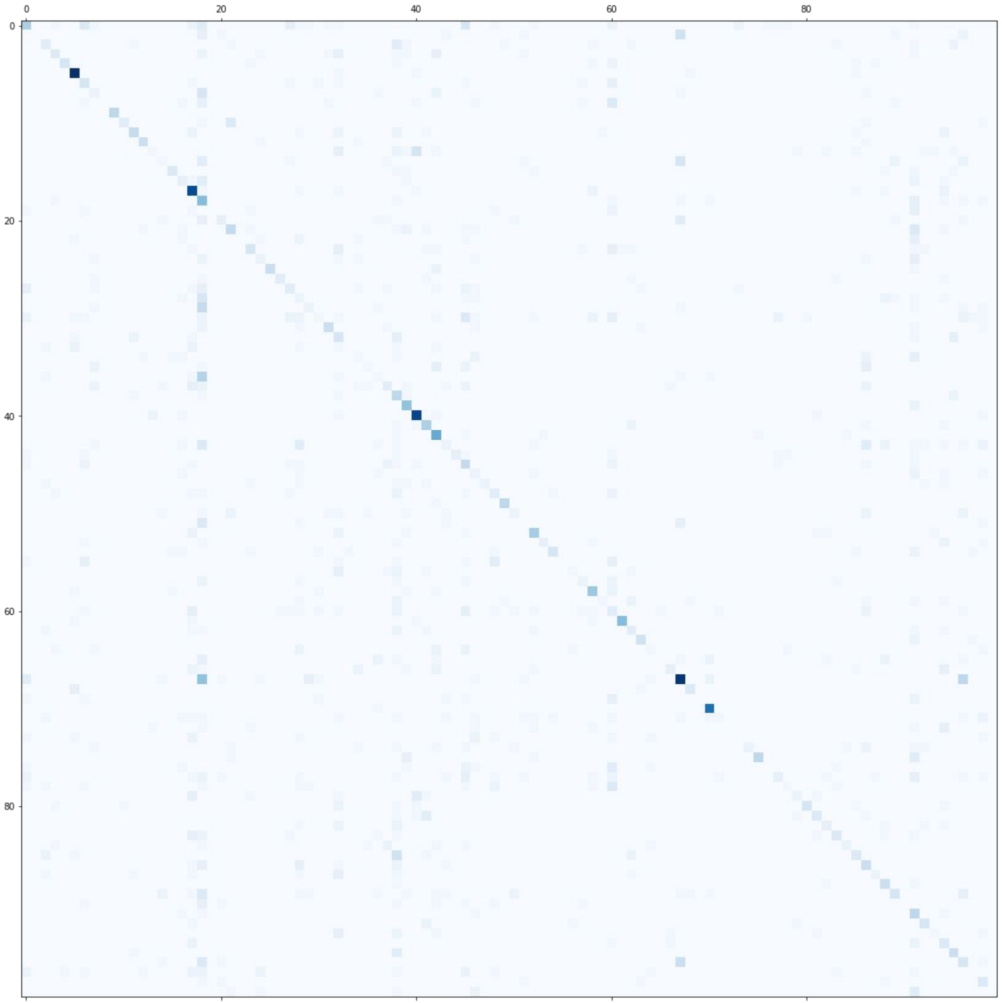
Accuracy top 1: 35.26%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 64.56%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 76.92%
Accuracy majority top 10 : 23.73%

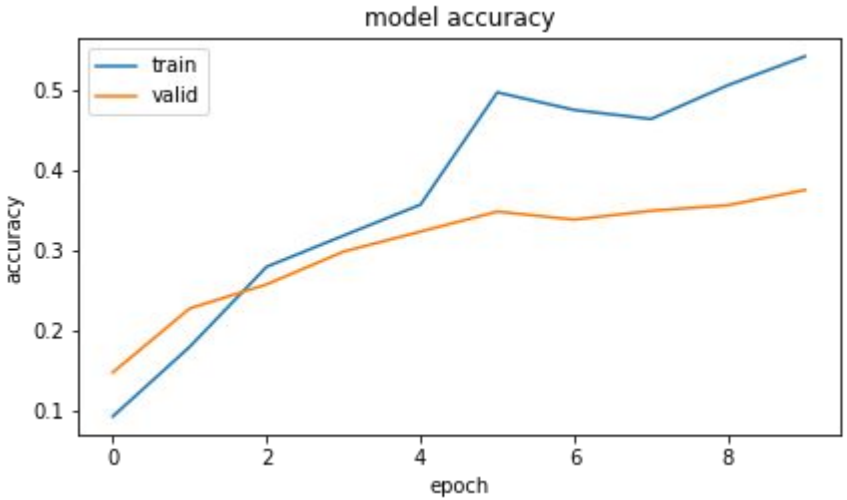
ResNet101
with rmsprop

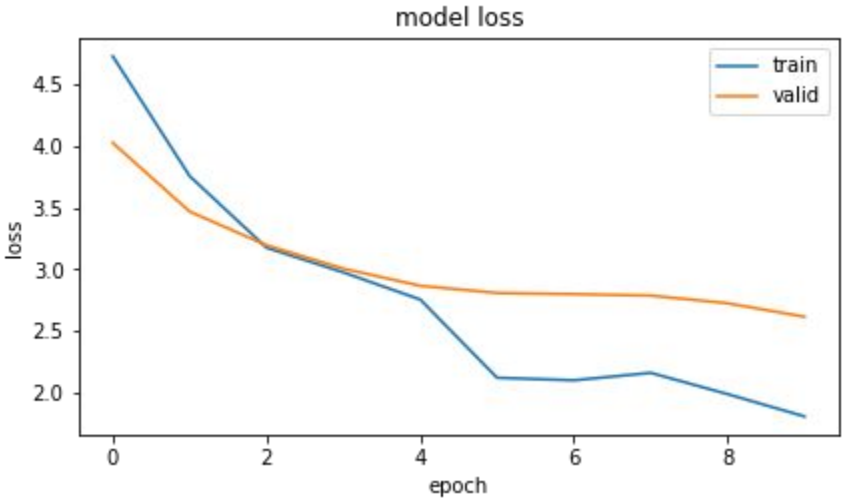
Confusion Matrix



ResNet101 with rmsprop

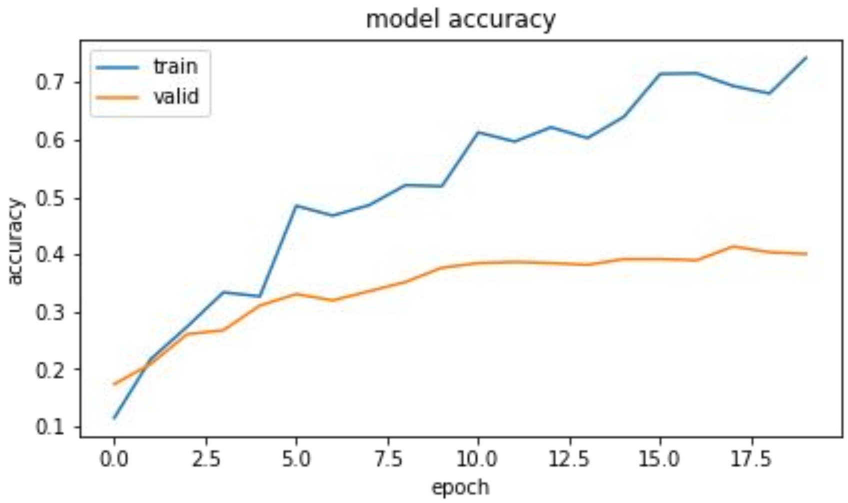
Loss seems to be decreasing -> increase epoch



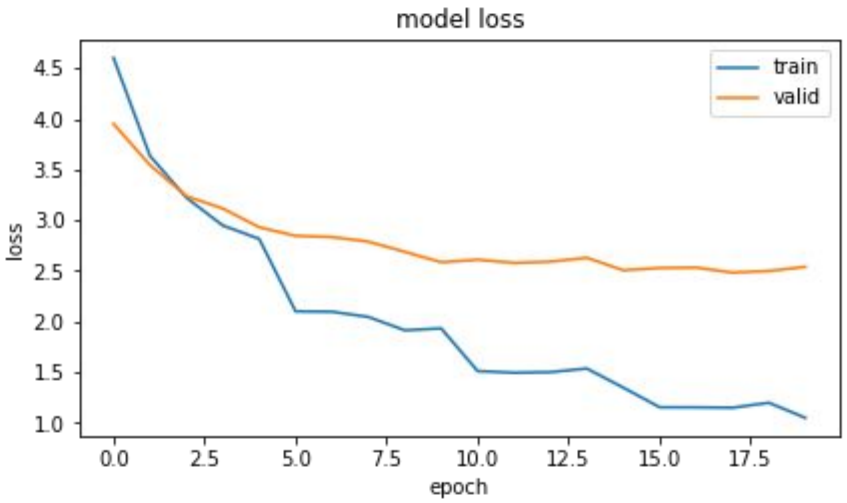


ResNet101 with rmsprop (20 epochs)

Accuracy on validation : 40.1%



model accuracy



model loss

Evaluate on test set(ResNet101 with rmsprop 20 epochs)

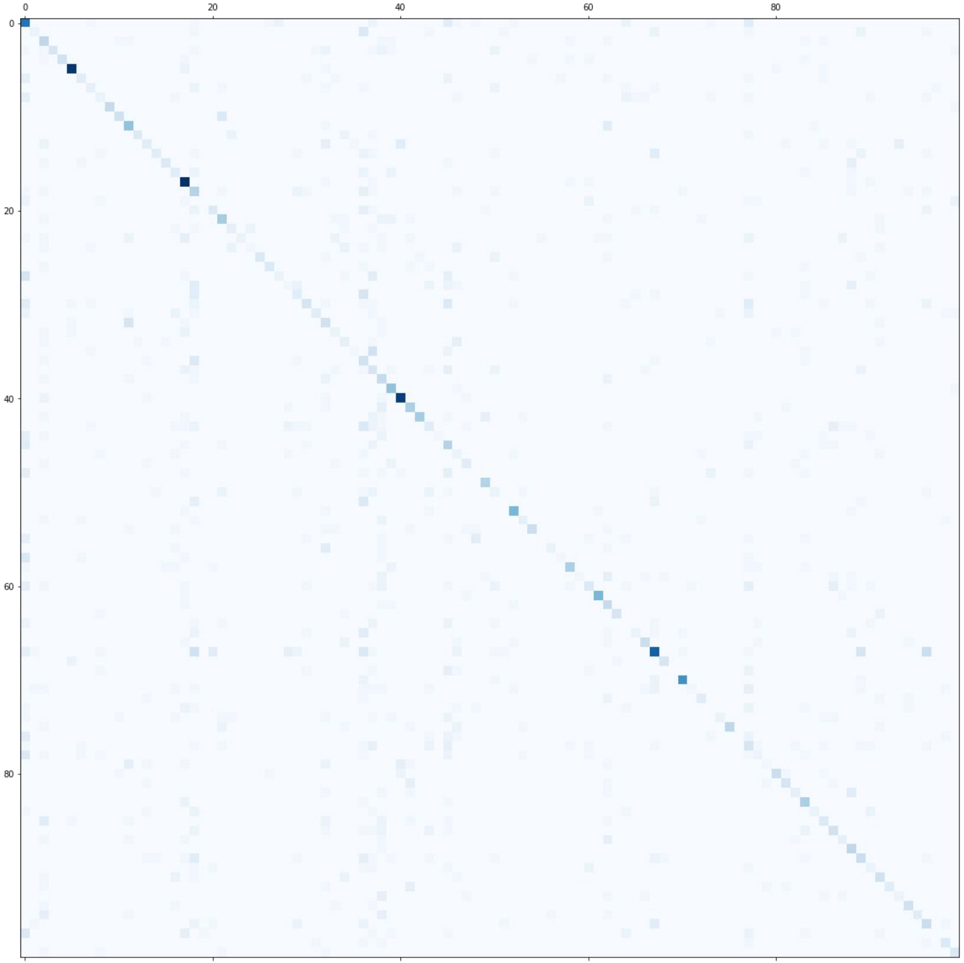
Accuracy top 1: 39.72%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 67.84%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 79.44%
Accuracy majority top 10 : 23.73%

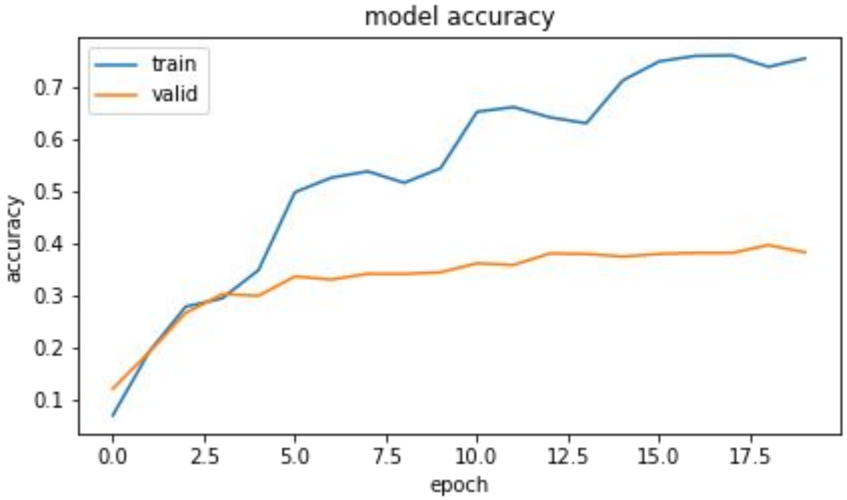
ResNet101 with rmsprop (20 epochs)

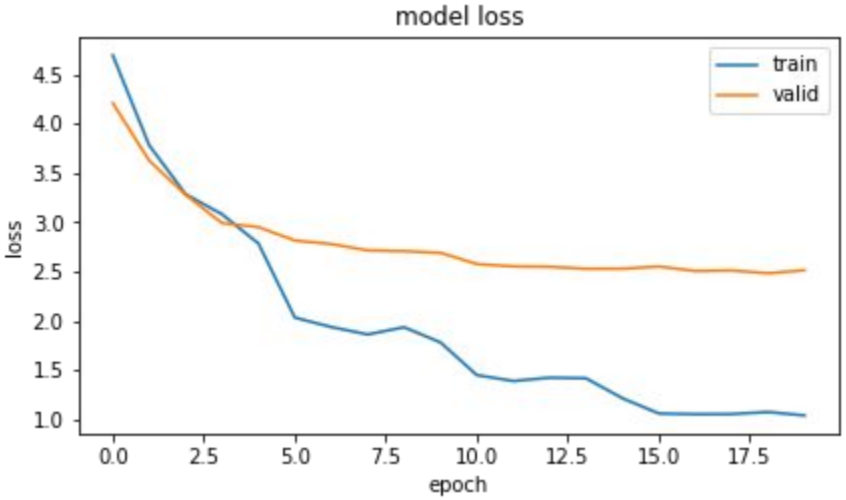
Confusion Matrix



ResNet101 with adam(20 epochs)

Accuracy on validation : 38.30%





Evaluate on test set(ResNet101 with adam 20 epochs)

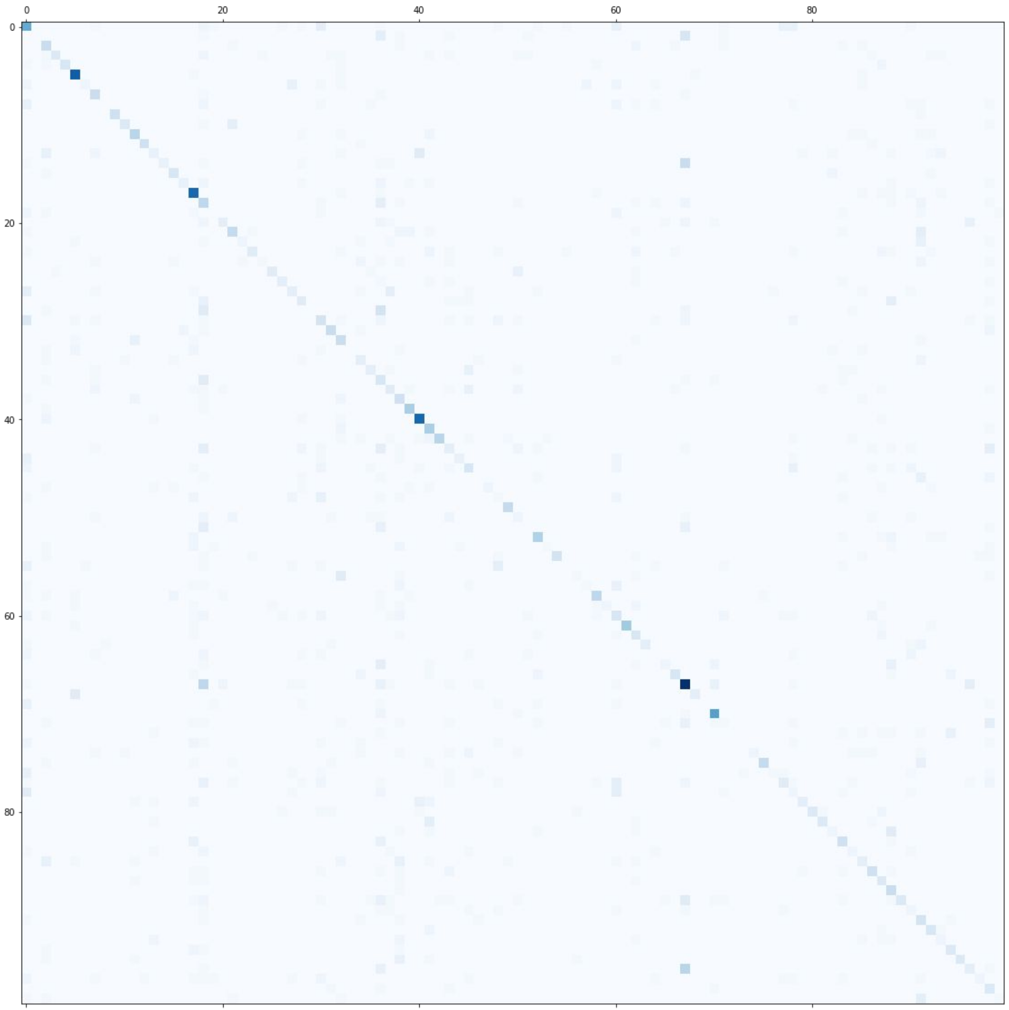
Accuracy top 1: 40.25%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 68.07%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 79.44%
Accuracy majority top 10 : 23.73%

ResNet101 with adam (20 epochs)

Confusion Matrix



Model Summary

Model	Accuracy Top 1	Accuracy Top 5	Accuracy Top 10
ResNet50 on test set	16.34%	33.45%	45.40%
ResNet101 on test set	17.98%	33.68%	46.16%
InceptionV3 w/ sgd	10.60%	20.74%	29.53%
InceptionV3 w/ rmsprop	29.23%	54.48%	69.83%
InceptionV3 w/ adam	28.06%	57.23%	69.83%
ResNet101 w/ rmsprop	35.26%	64.56%	76.92%
ResNet101 w/ rmsprop 20 epochs	39.72%	67.84%	79.44%
ResNet101 w/ adam 20 epochs	40.25%	68.07%	79.44%

Dataset Cleaning

VE VANT YOUR BLOOD!

Leesylvania State Park is looking for individuals willing to volunteer their veins for a blood drive on October 27th to help the American Red Cross save lives.

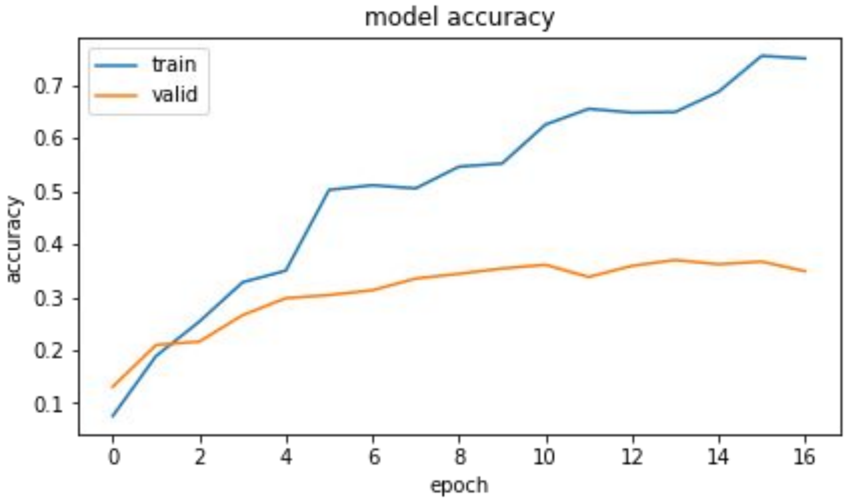
BUT...we need 40 people to sign up to surrender to the syringe before the Red Cross will agree to bring all of their equipment and staff to the park .

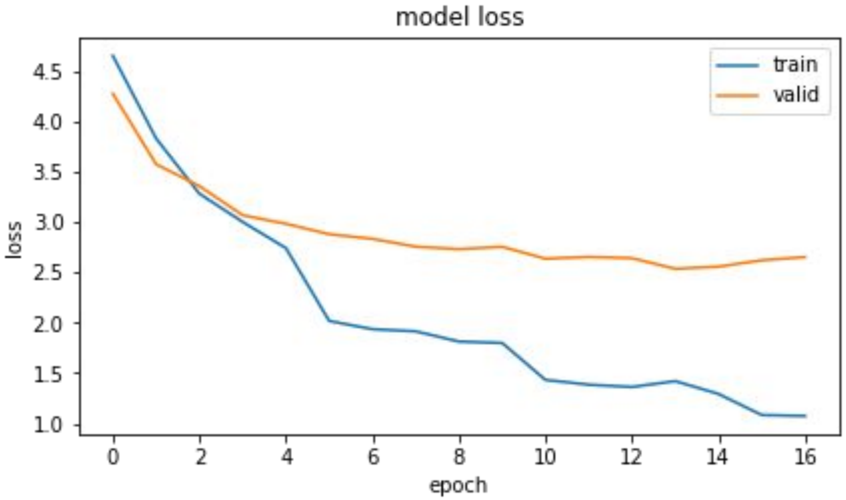
Become a local hero (cape not included) and sign up at the reception desk in the Visitor Center or call 703-583-6904.

American Red Cross

ResNet101 with adam(20 epochs) and cleaned dataset

Accuracy on validation : 35.0%





Evaluate on test set(ResNet101 with adam 20 epochs and cleaned dataset)

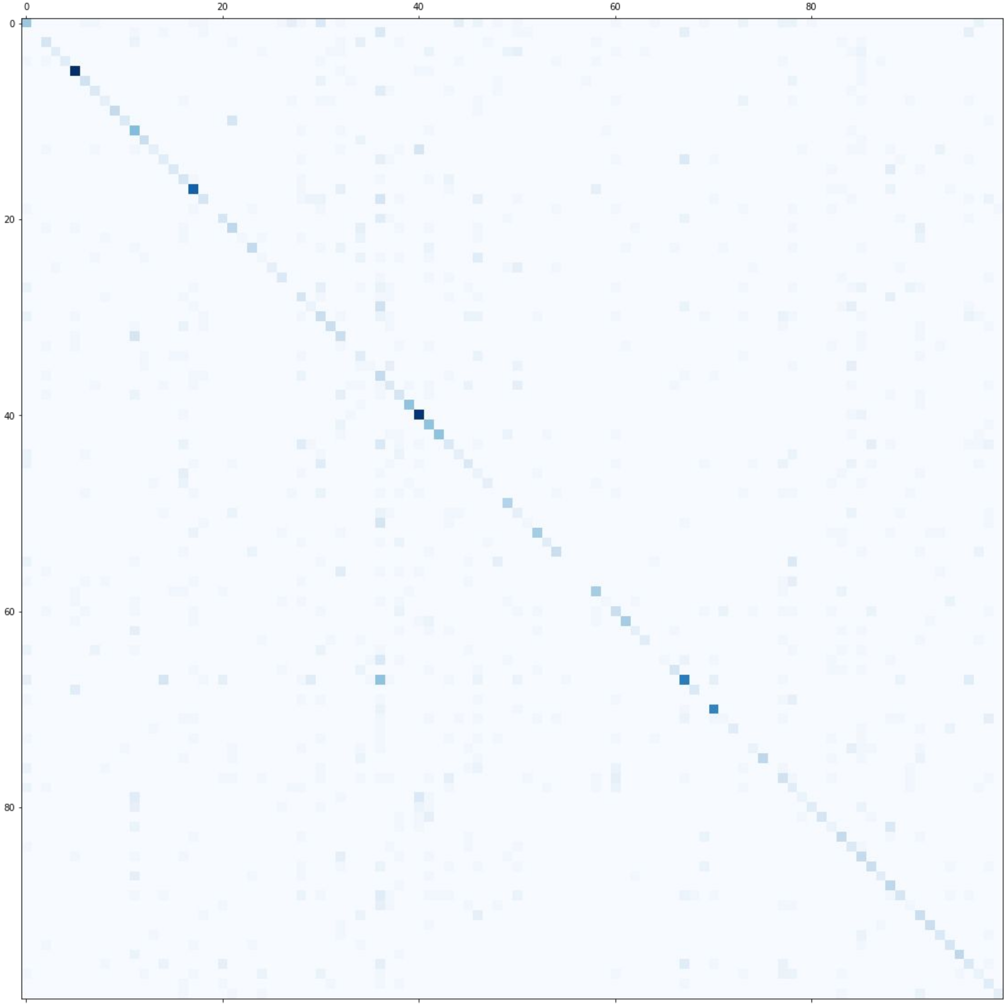
Accuracy top 1: 38.25%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 68.84%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 79.50%
Accuracy majority top 10 : 23.73%

ResNet101 with adam and cleaned dataset (20 epochs)

Confusion Matrix



Summary

- Adam and rmsprop is more preferable than sgd in this case.

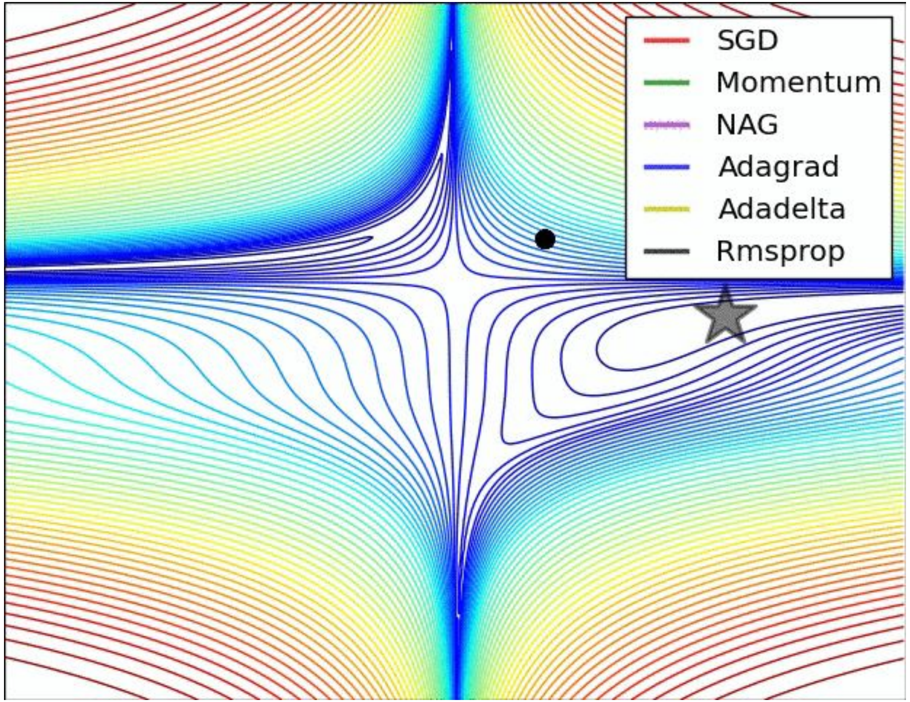
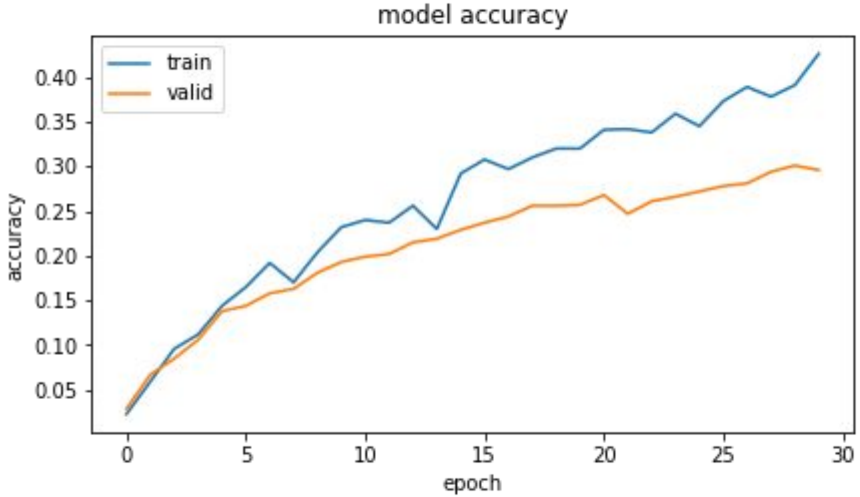
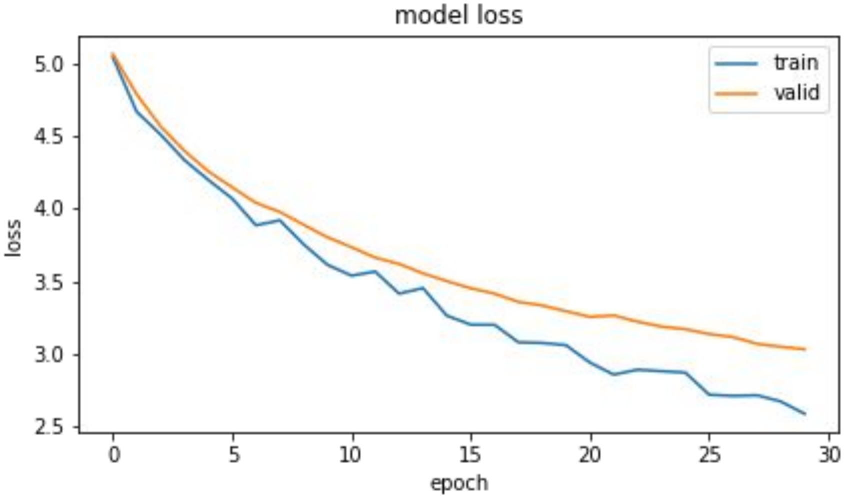


Image Credit: https://cs231n.github.io/neural-networks-3/

ResNet101 with sgd(30 epochs)

Accuracy on validation : 29.6%





Evaluate on test set(ResNet101 with sgd 30 epochs and cleaned dataset)

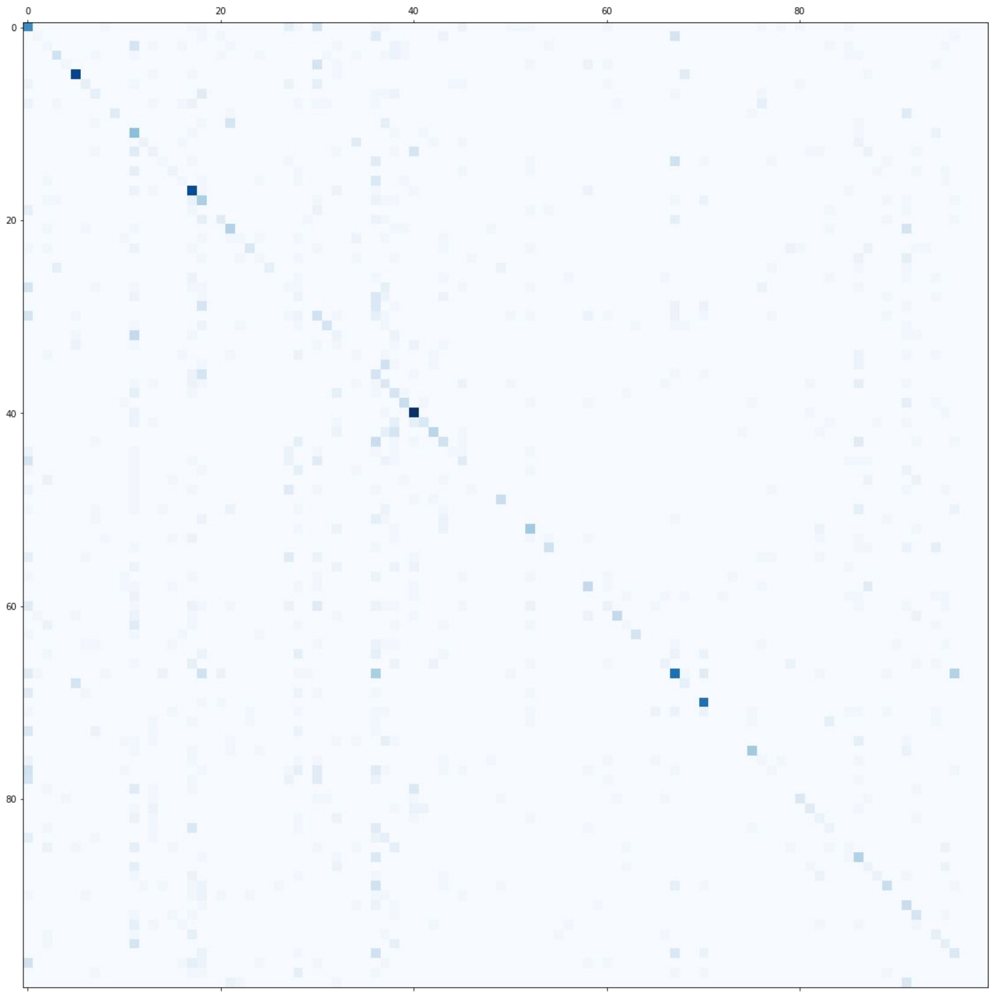
Accuracy top 1: 27.30%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 54.19%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 66.20%
Accuracy majority top 10 : 23.73%

ResNet101 with sgd (30 epochs)

Confusion Matrix



ResNet101(sgd) 10 vs 30 epochs

10 epochs	30 epochs
Accuracy top 1: 17.98% Accuracy majority top 1 : 4.73%	Accuracy top 1: 27.30% Accuracy majority top 1 : 4.73%
Accuracy top 5 : 33.68% Accuracy majority top 5 : 15.10%	Accuracy top 5 : 54.19% Accuracy majority top 5 : 15.10%
Accuracy top 10 : 46.16% Accuracy majority top 10 : 23.73%	Accuracy top 10 : 66.20% Accuracy majority top 10 : 23.73%

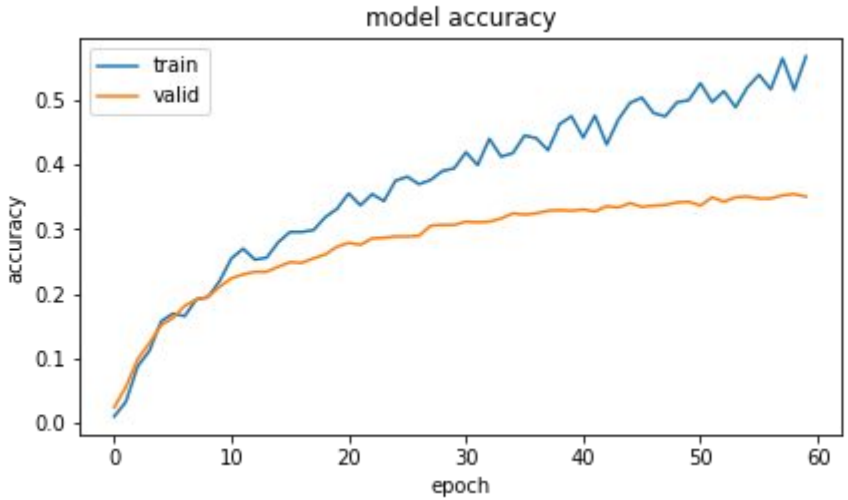
Let’s try 100 epochs(with callback)

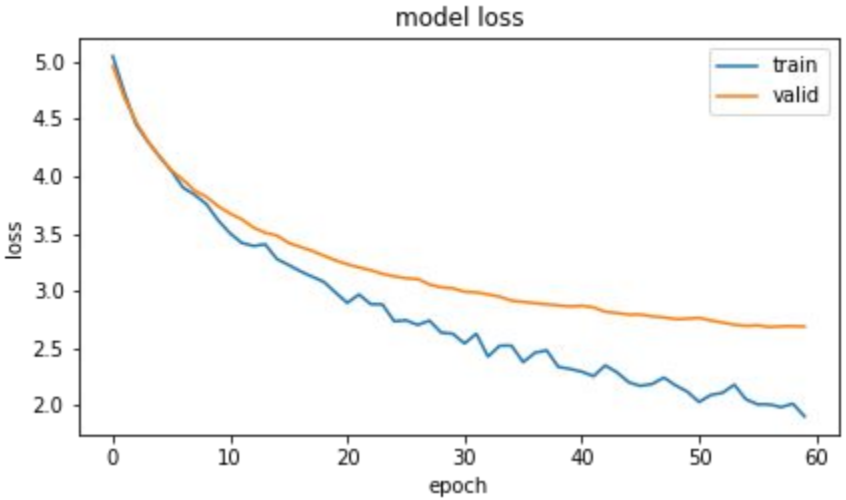
But...

our callback function stop at the 60th epoch.

ResNet101 with sgd(60 epochs)

Accuracy on validation : 35.1%





Evaluate on test set(ResNet101 with sgd 60 epochs and cleaned dataset)

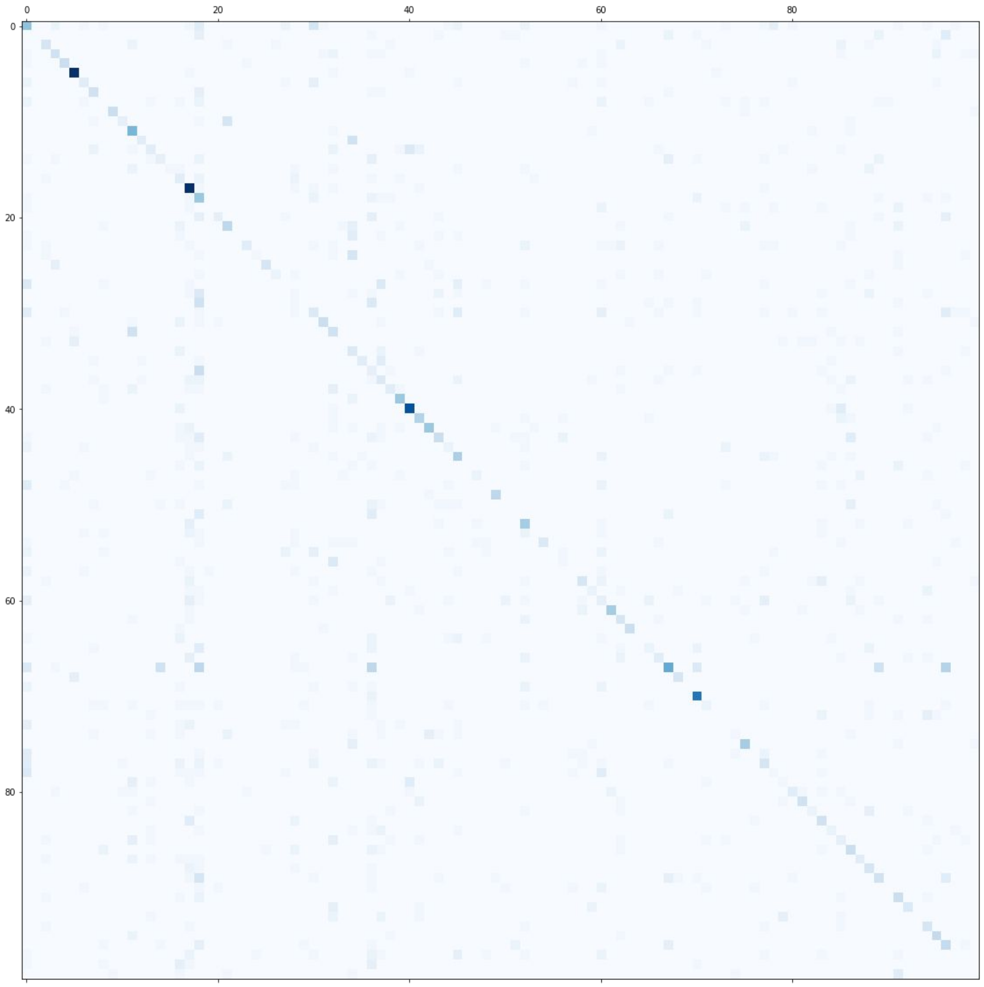
Accuracy top 1: 33.86%
Accuracy majority top 1 : 4.73%

Accuracy top 5 : 60.98%
Accuracy majority top 5 : 15.10%

Accuracy top 10 : 74.17%
Accuracy majority top 10 : 23.73%

ResNet101 with sgd (60 epochs)

Confusion Matrix



ResNet101(sgd) 30 vs 60 epochs

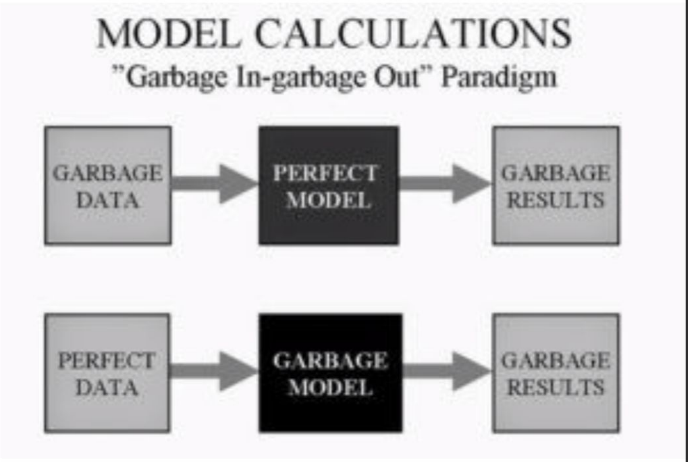
30 epochs	60 epochs
Accuracy top 1: 27.30% Accuracy majority top 1 : 4.73%	Accuracy top 1: 33.86% Accuracy majority top 1 : 4.73%
Accuracy top 5 : 54.19% Accuracy majority top 5 : 15.10%	Accuracy top 5 : 60.98% Accuracy majority top 5 : 15.10%
Accuracy top 10 : 66.20% Accuracy majority top 10 : 23.73%	Accuracy top 10 : 74.17% Accuracy majority top 10 : 23.73%

sgd vs adam vs rmsprop on ResNet101

optimizer/epochs	Accuracy top1	Accuracy top5	Accuracy top10
rmsprop/10	35.26	64.56	76.92
rmsprop/20	39.72	67.84	79.44
adam/20	40.25	68.07	79.44
adam/20 cleaned	38.25	68.84	79.50
sgd/30 cleaned	27.30	54.19	66.20
sgd/60 cleaned	33.86	60.98	74.17

Garbage in Garbage out

- The machine is as intelligent as the data/features we put in
- “Garbage in, Garbage out”
- Data cleaning is often done to reduce unwanted things



Ref: https://github.com/ekapolc/pattern_2019

https://precisionchiroco.com/garbage-in-garbage-out/