Multi-class Image Classification using Deep Convolutional Networks on extremely large dataset

Marija Stanojevic
Ashis Chanda
CIS Department, Temple University
5th December 2017

Content

- Our Problem
- Dataset
- Proposed Models
- Challenges
- Experimental Result
- Conclusions
- Future work

Problem Definition

- Kaggle competition: **Cdiscount's Image Classification Challenge**
- Classify user posted images into 5270 categories
- Challenges of dataset:
  - Huge amount of data and categories
  - Background clutter (objects blend into environment)
  - Viewpoint and data scale variation
  - Deformation
  - Occlusion
  - Illumination conditions
- Deep learning models seem suitable

Category	**# Labels**
Category1	49
Category2	483
Category3	5270

Data sets	**Size**	**# products**
Train data	58.2 GB	7,069,896
Test data	14.5 GB	1,768,182

Dataset: Product complexity

Figure 1. Deformation and scaling problem in product images

Figure 2. Occlusion problem in product images

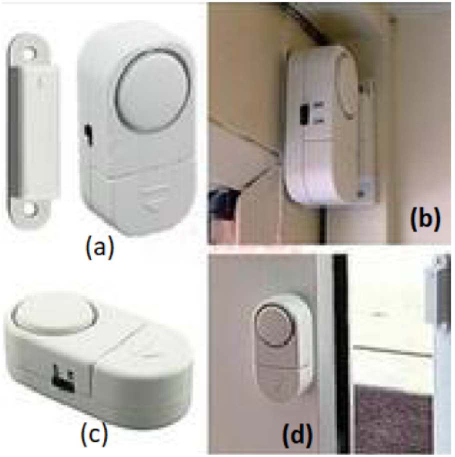


Figure 3. One item in different angles and backgrounds

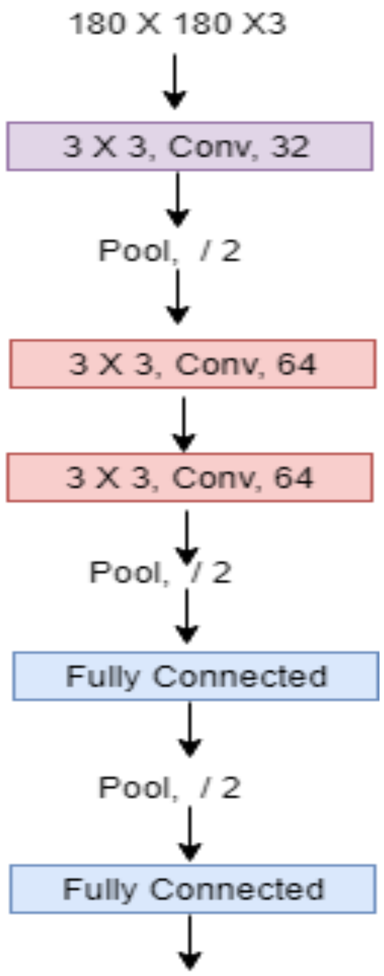
Dataset: Categories

A sample list of product categories:

category_id	category_level1	category_level2	category_level3
1000010629	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	ADAPTATEUR DE CARTE SIM
1000010631	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	AMPLIFICATEUR D'APPEL TELEPHONIQUE
1000010633	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	AMPLIFICATEUR DE SIGNAL - ANTENNE
1000019772	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BATTERIE EXTERNE - POWER BANK POUR
1000010635	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BATTERIE TELEPHONE
1000010637	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BIJOU DE TELEPHONE
1000010639	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BOUCHON ANTI-POUSSIERE
1000019766	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BRASSARD DE MARCHE POUR TELEPHONE
1000010641	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	CABLE TELEPHONE
1000010643	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	CARTE SIM
1000010645	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BLUETOOTH

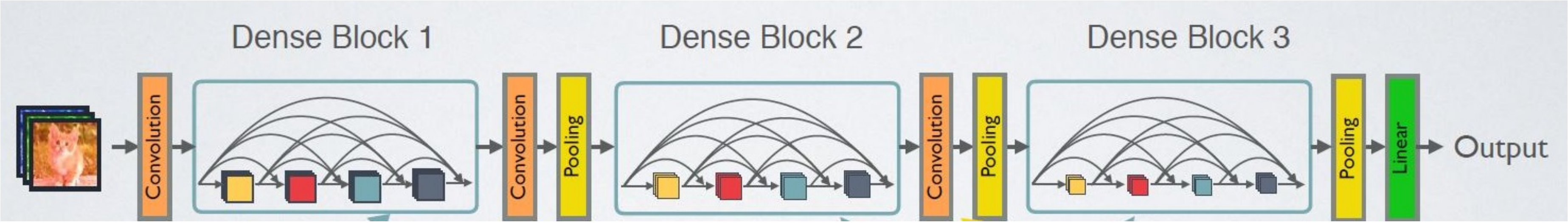
Models: CNN

- **Our CNN**
  - 3 Conv, 2 FC layers, 3 Pool layers
  - thinner layers
  - downsampling three times
- Tried to use for three types of categories



Models: DenseNet

- l th layer has connection with all previous layers (l -1)
- Normally, each dense block contains 40, 100, 160 layers
- We used 3 blocks, 3 layers. Total connection: l x (l -1) /2 = 15
- Each layer has BN, ReLU, Conv
- No transition layer (downsampling), but used dropout



Models: ResNet

- ❖ 34, 50, 101, 152 layers
- ❖ Residual block example
- ❖ Batch normalization, ReLu

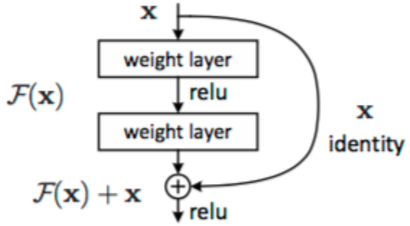
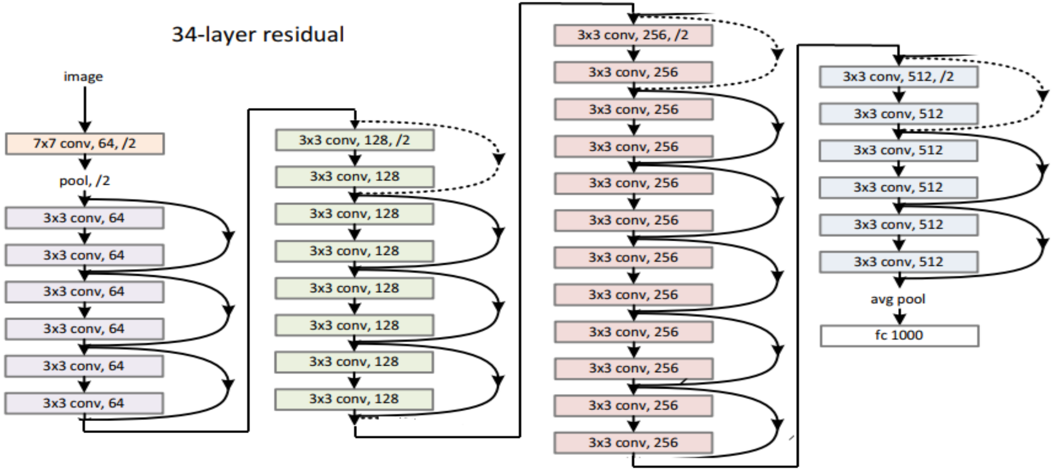
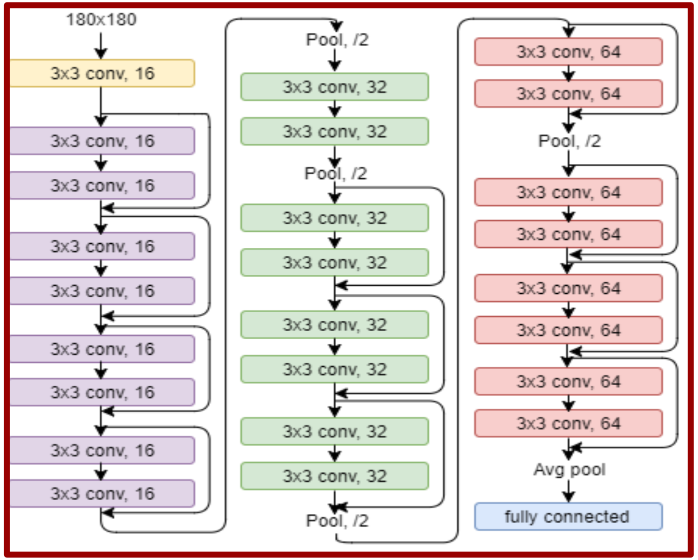


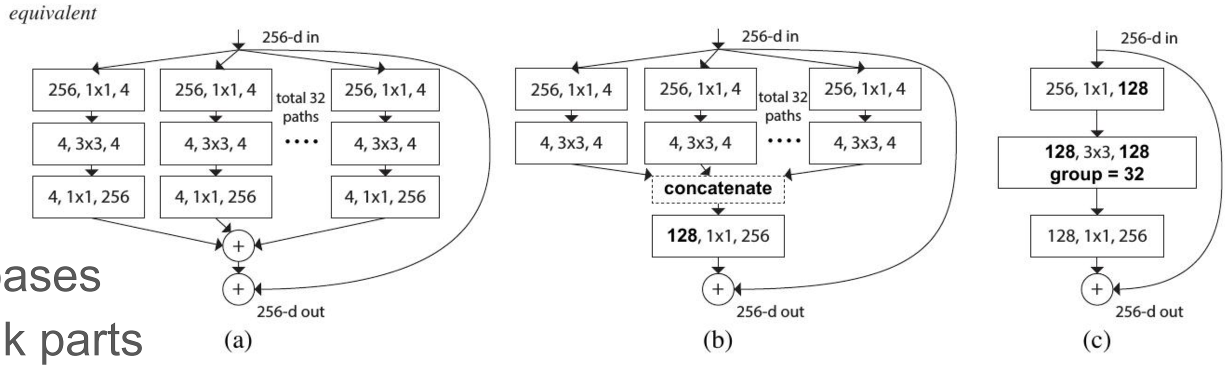
Figure 2. Residual learning: a building block.

- ❖ Our residual network
  - ➢ thinner layers (25 conv + 4 pool + fc)
  - ➢ downsampling after each block
  - ➢ no bottleneck

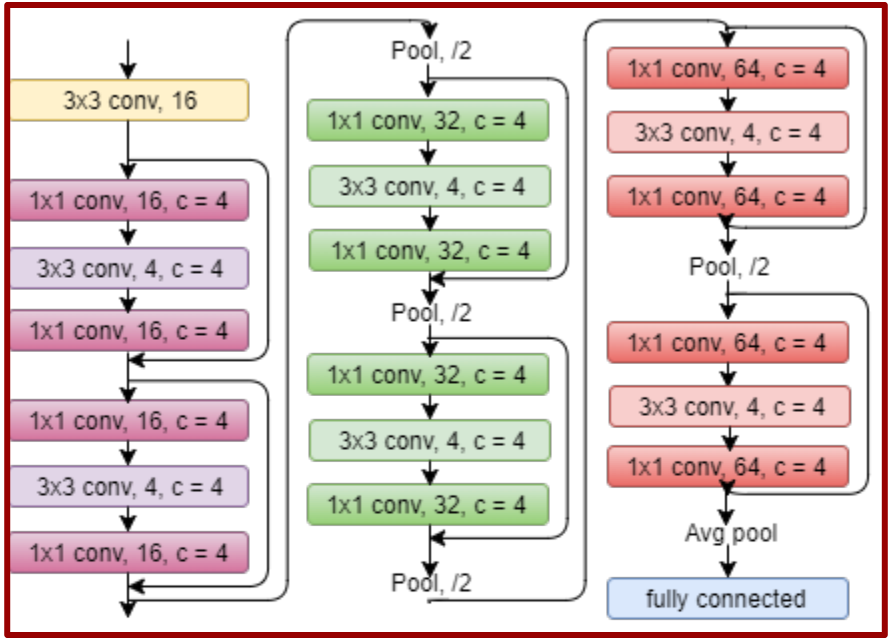




Models: ResNext

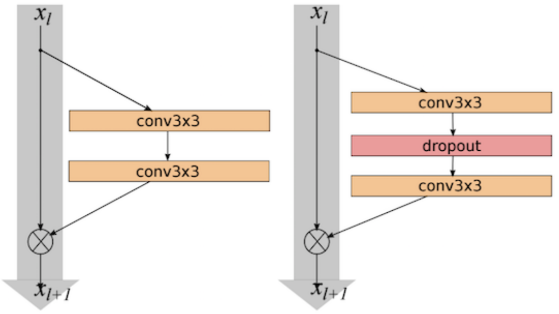


- Uses residual block as bases
- Block width is divided in k parts which learn separately
- Blocks are wider than in resNet
- Distinct cardinalities
- 29, 50 and 101 layers
- **Our resNext**:
  - thinner layers
  - downsampling after each block
  - less layers (19 conv + 4 pool + fc)
  - cardinality 32 always - best results according to paper

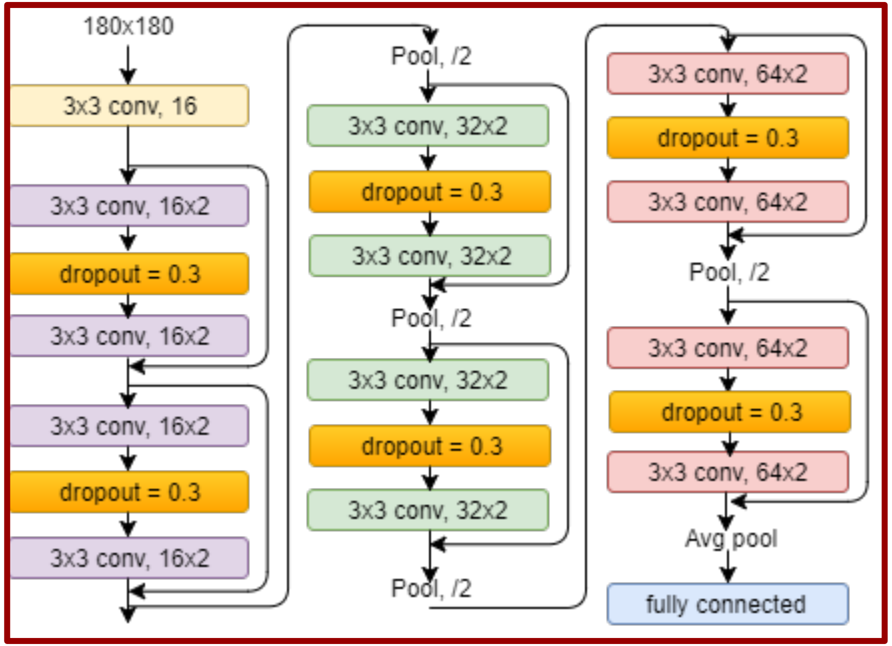


Models: WideResNet

- Uses residual block as bases
- Block is k times wider (k=1, 2, 4, 8, 10, 12)
- Dropout beside batch normalization
- 16, 22, 28, 40 layers
- **Our wideResNet**:
  - k = 2 - doesn’t increase much number of parameters, but shows biggest improvement in original results
  - dropout keep = 0.7 - best in original results
  - less layers (13 conv + 4 pool + fc)



(c) basic-wide (d) wide-dropout



Challenges in dataset

- 5.6 TB after splitting data
- Train data (707 Batches) & Test data (177 Batches)
- **Smaller batch**: more reads and writes
- **Bigger batch**: memory error
- Each batch has 10,000 products ~ 20,000 images
- Cross-validation data: 707 products ~ 1500 images

Challenges in implementation

- ❖ Used **owlsnesttwo** high performance computing system
- ❖ GPU: NVIDIA Tesla P100 PCIe 12GB
- ❖ Only two jobs allowed in parallel
- ❖ Implemented with tensorflow and tflearn libraries in python
- ❖ Network
    - ➢ Complexity in debugging
    - ➢ **Bigger network**: memory error => tuning network
    - ➢ Thinner and less layers
    - ➢ Small number of epoch

Baseline (CNN) Result

- **CNN: (10 Epochs, 10 Batches)**

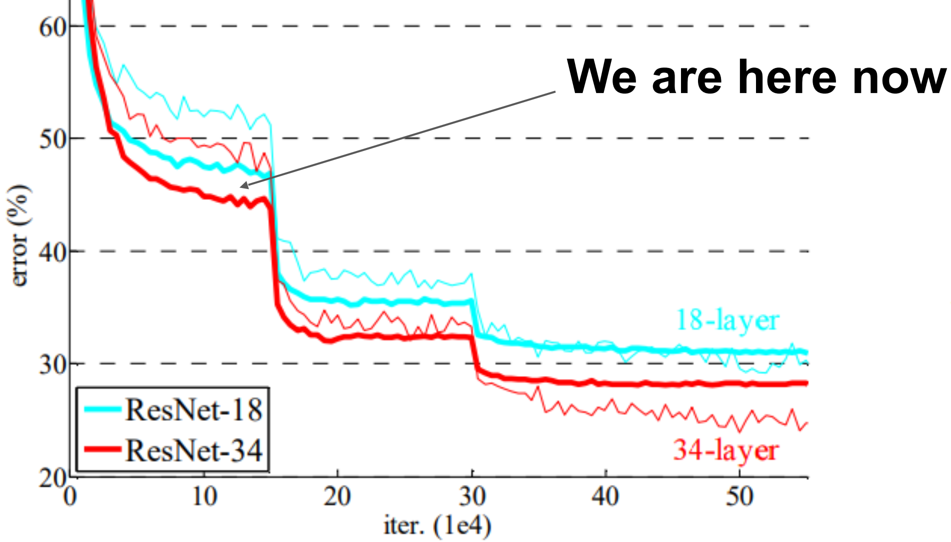
Label	# class	Accuracy	Error	Time
Category 1	49	52%	10.7	4 hours
Category 2	483	47%	11.68	5 hours
Category 3	5270	32%	4.89	11 hours

- **CNN: (50 Epochs, 50 Batches) (in progress)**

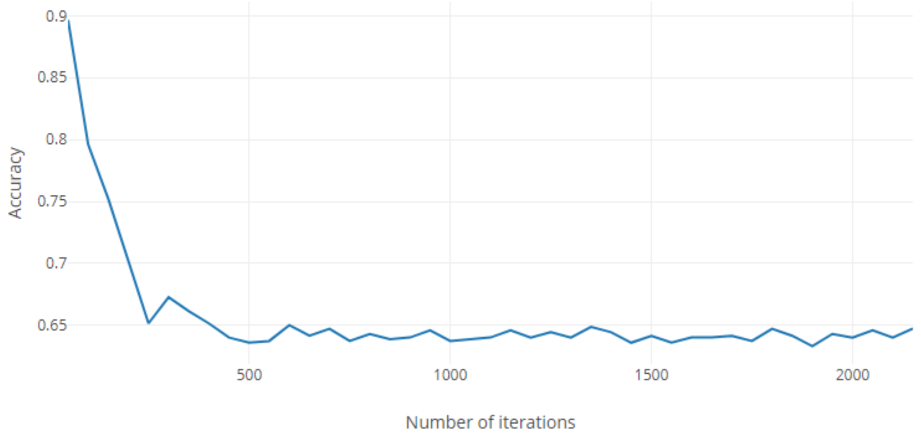
Category 1	49	12% (1 epoch)	3.42	20 hours

Experimental Result (category 3)

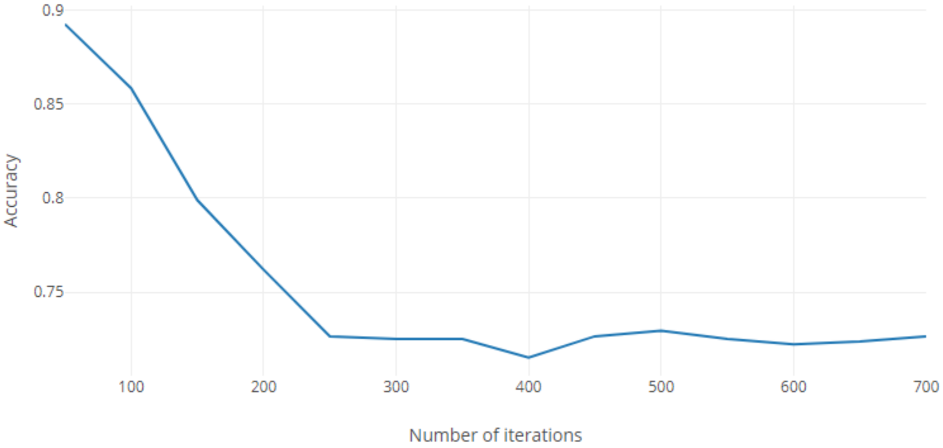
Model	# Batch	# Epoch	Accuracy	Error	Time
DenseNet	50/707	1 (running)	4%	9.83	10 days
ResNet	50/707	10	34%	3.94489	5.69 hours
ResNext	50/707	10	28.9%	4.51031	16.8 hours
WideResNext	50/707	10	**41.93%**	3.53318	6.11 hours
ResNet	707/707	3 (running)	36.69%	3.84427	3.35 days
ResNext	707/707	1 (running)	28.47%	4.50394	9.90 days
WideResNext	707/707	2 (running)	**40.51%**	3.49804	3.60 days



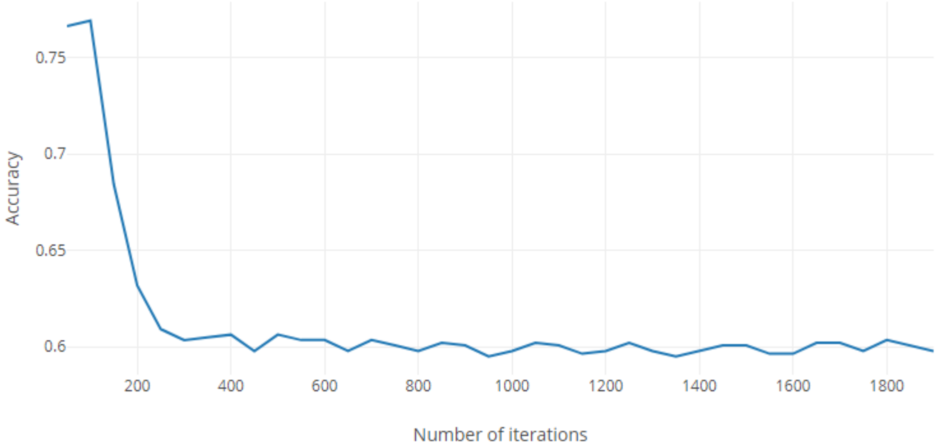
Accuracy of resNet on all data (707 batches)



Accuracy of resNext on all data (707 batches)



Accuracy of wideResNet on all data (707 batches)



Experimental results: Error in classifying

Mobile cover bag

Mobile Case

Laptop cover bag Case

Mobile film protector

Mobile

Conclusions and Future Work

- **Baseline:** CNN; **Proposed:** resNet, resNext, denseNet, wideResNet
- All proposed networks have similar number of parameters
- wideResNet performs the best
- resNext gave worst results and is 3 times slower than resNet
- DenseNet requires GPU with huge memory
- Requires a lot of time, huge memory and fast computational resources
- Number of epochs has to be 70+
- **Future:** Submit result to Kaggle competition

Thank you

Questions?

Baseline (CNN) Result

- **CNN: (10 Epochs, 10 Batches)**

Label	# class	Accuracy	Error	Time
Category 1	49	52%	10.7	4 hours
Category 2	483	47%	11.68	5 hours
Category 3	5270	32%	4.89	11 hours

- **CNN: (50 Epochs, 50 Batches) (in progress)**

Category 1	49	12% (1 epoch)	3.42	20 hours

Model	# Batch	# Epoch	Accuracy	Error	Time / 10 epochs
CNN	50/707	10	32%	4.89	19 hours
DenseNet	50/707	10	16.74%	5.38	1.5 days
ResNet	50/707	10	34%	3.94489	5.69 hours
ResNext	50/707	10	28.9%	4.51031	16.8 hours
WideResNet	50/707	10	**41.93%**	3.53318	6.11 hours
CNN	50/707	50	**38.23%**	4.056	19 hours
DenseNet	50/707	50	22.73%	4.04	1.5 days
ResNet	707/707	25 (running)	37.25%	3.99638	3.35 days
ResNext	707/707	3	28.75%	4.50174	9.90 days
WideResNe	707/707	20	**41.64%**	3.50708	3.60 days