Geez Character Recognition Using Deep Convolutional Neural Network

Tesfay Gebrekirstos

Addis Ababa Universty
Addis Ababa Institute of Technology
Master's Degree in Communication Engineering

Thesis Defense
May 17, 2019

Advisor: Alhayat Ali(PhD)
Co-advisor: Ephrem Teshale(PhD)

Outline

Back ground

OCR is the conversion of scanned Printed documents, handwritten documents and images to digital form so that it can be searched and indexed.

idea

- Exams are corrected manually and scores marked on the first sheet.
- scores entered in to online portal.
- want to automate the mark entry
  - just scanning the first sheet will pick student ID and marks and save in an electronic form.

Statement of the Problem

One of the biggest challenges working with contextualized apps in Ethiopia is the lack of existing supportive language tools. Creating a Tigrigna or Amharic based system usually lacks seamless Geez writing tool that contains integrated grammar and spelling checker. How about working with Geez characters in the Big data scenario? Coming to our case, our big data is not in a digital format. Our big data is in an old printed documents and old mans. Developing Geez OCR and preparing Geez character datasets for future works is a primary step to achieve but has had limited success.

Objective of the Thesis

- To design and implement Geez character recognition system using deep learning algorithms
- To introduce Geez character datasets for future works.

Machine Learning and Deep Learning

Machine Learning

Machine learning refers to the use of algorithms to parse data, process and learn from it, in order to make predictions or determinations about something.

- One of the best application for machine learning is computer vision: OCR, object tracking, object recognition etc.

Deep Learning

Deep learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks (ANNs).

- Compared to older ML algorithms, Deep Learning performs better with a large amount of data.

Machine Learning And Deep Learning

- In the past 10 years, machine learning and Artificial Intelligence have shown tremendous progress.
- The recent success can be attributed to:
  - Emerging of data science /big data
  - Cheap computing cost – CPUs and GPUs
  - Improvement of machine learning models
- Much of the current excitement concerns a subfield of it called deep learning.

Machine Learning And Deep Learning

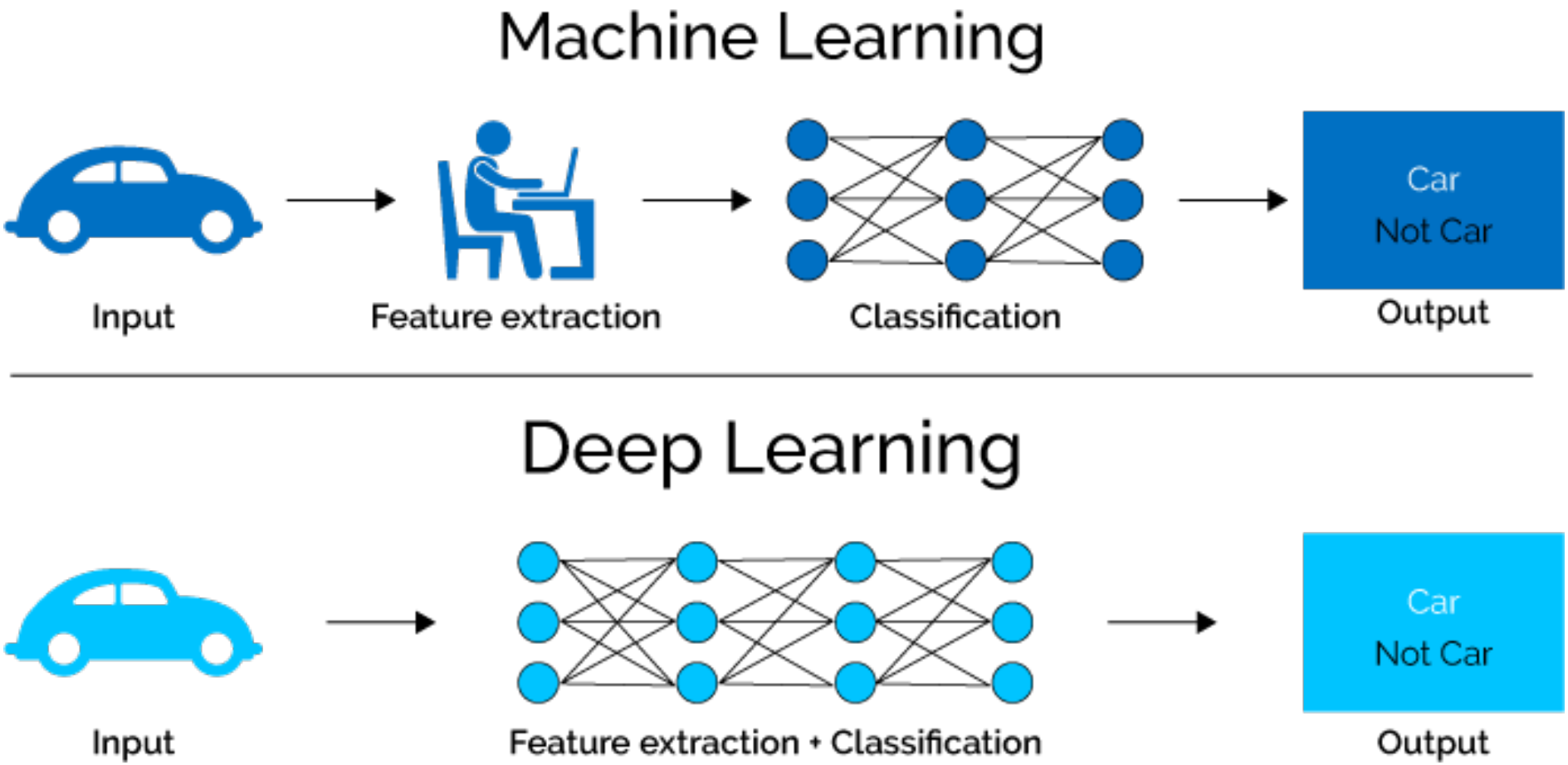
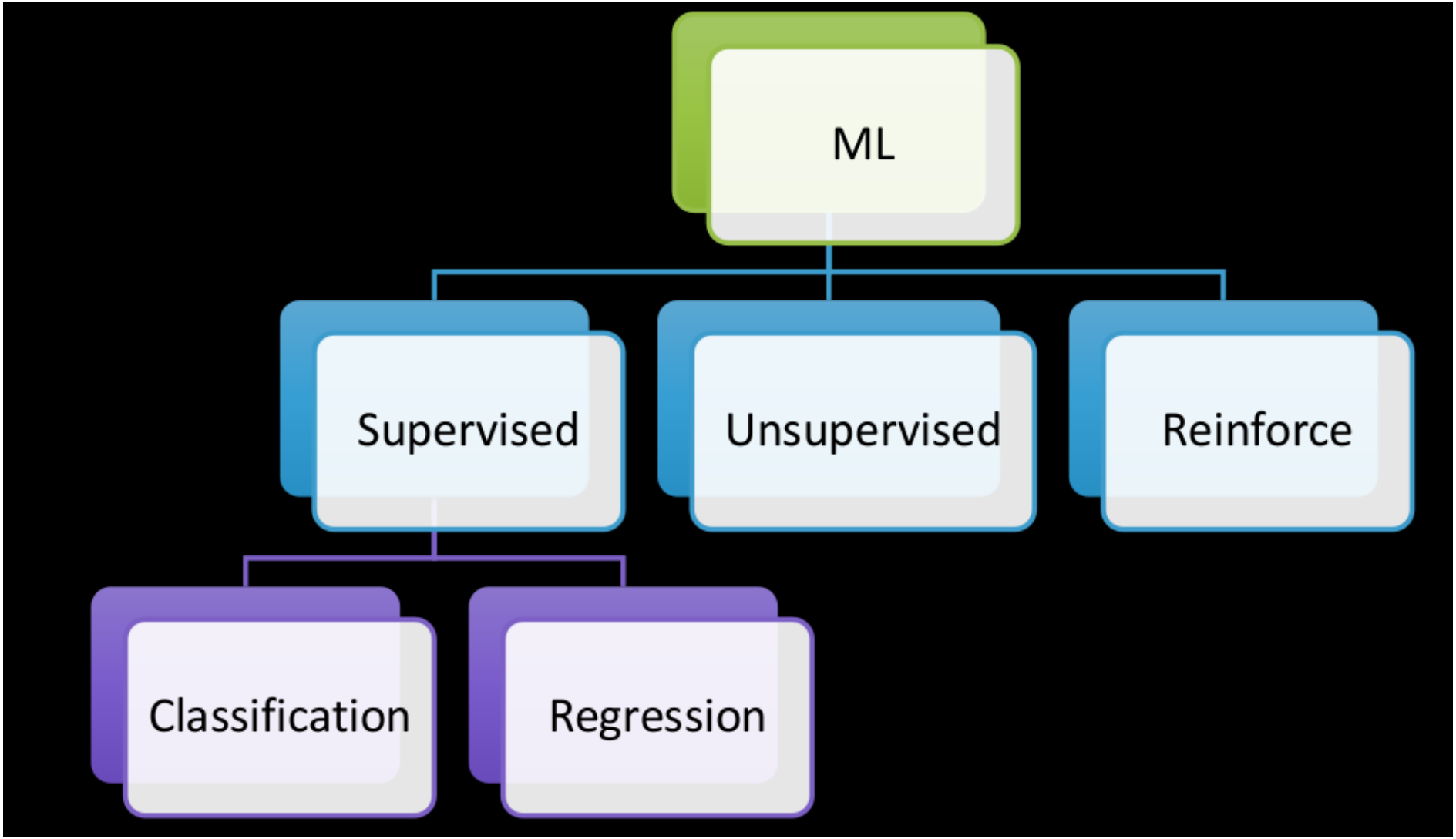


Figure: classical Vs Deep learning

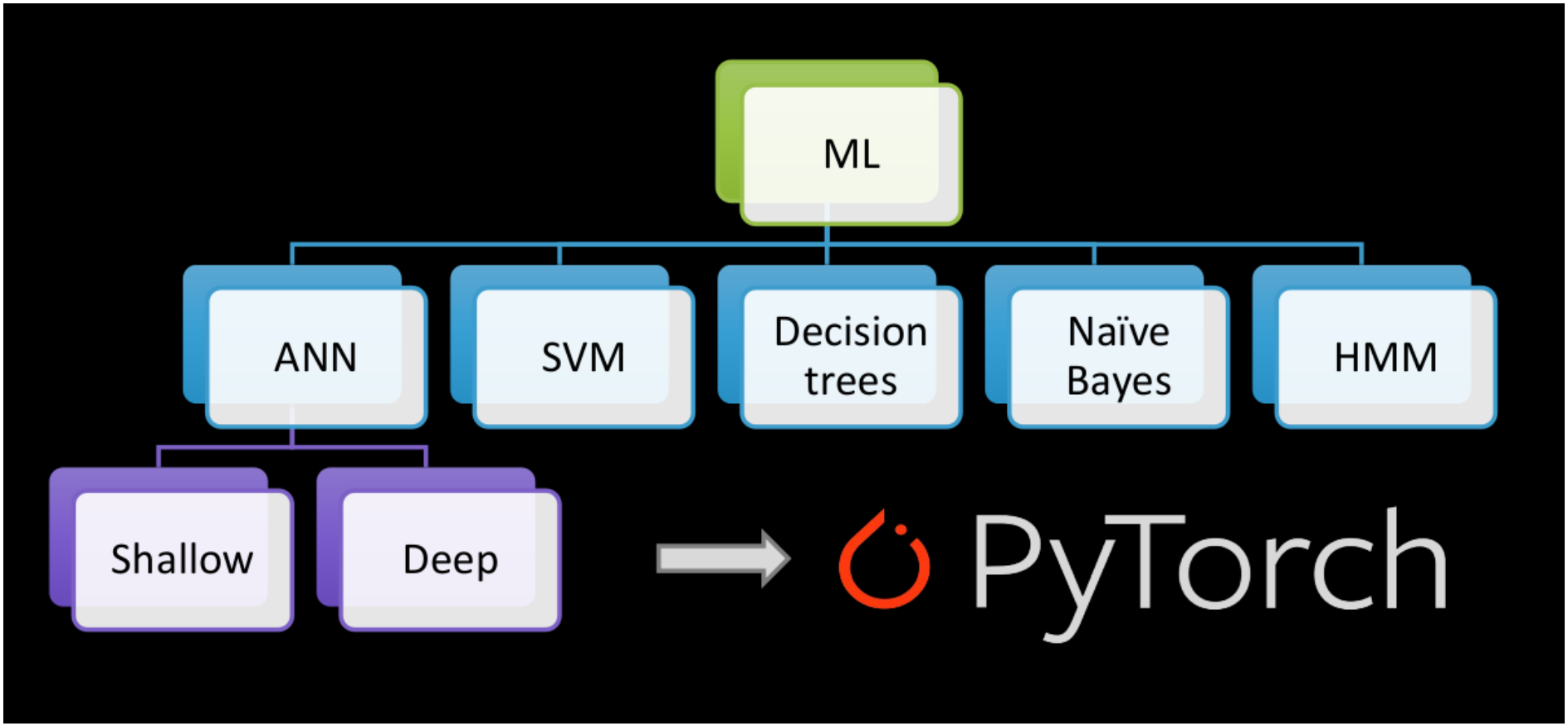
Taxonomy of ML

Learning paradigms and algorithms



Taxonomy of ML Algorithms

Learning paradigms and algorithms



Deep Learning: Application

Figure: Application area of Deep learning

Image classification

- Image classification can get really hard.

- CNNs represent current state-of-the-art technique in image classification, object detection etc.

Image classification

ILSVRC:

- 1,000 object classes
- 1,431,167 images

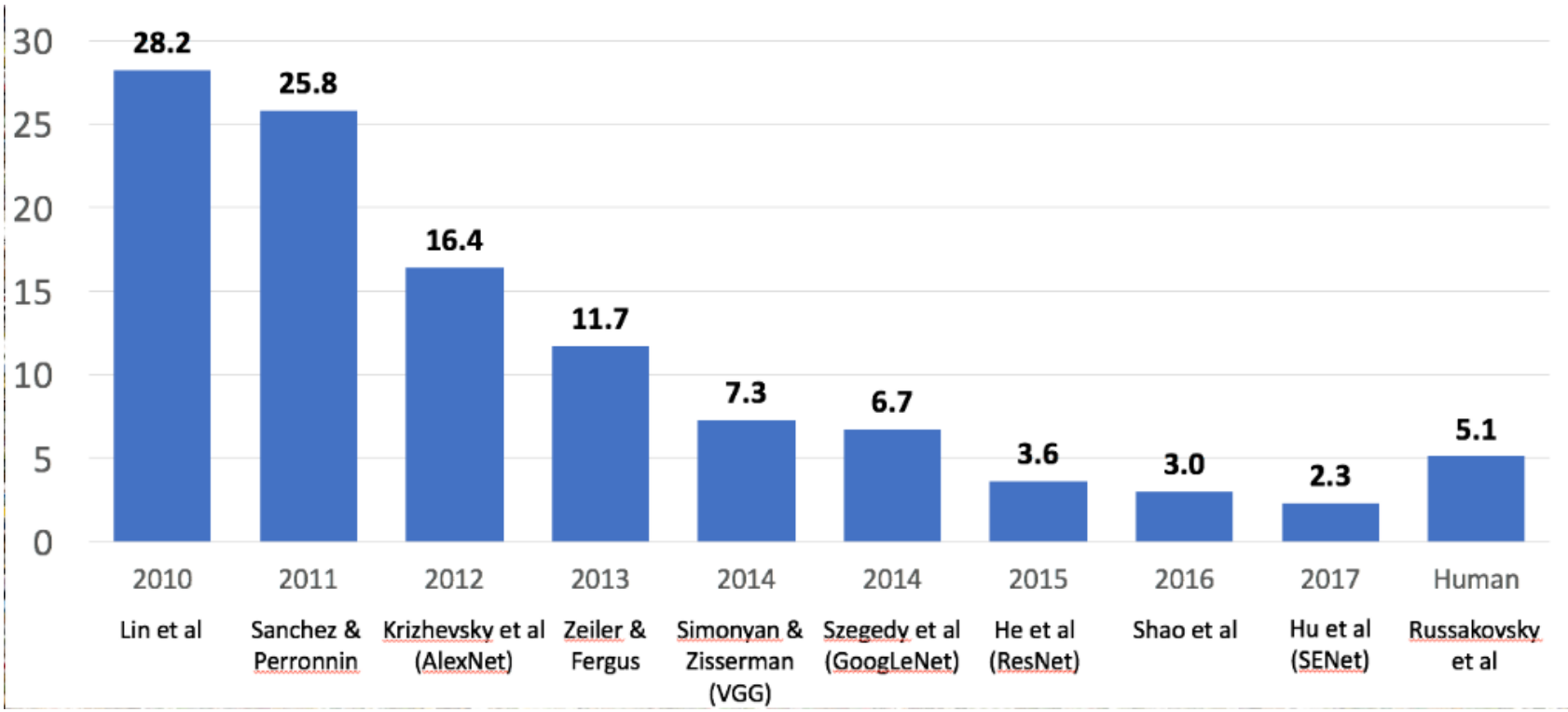
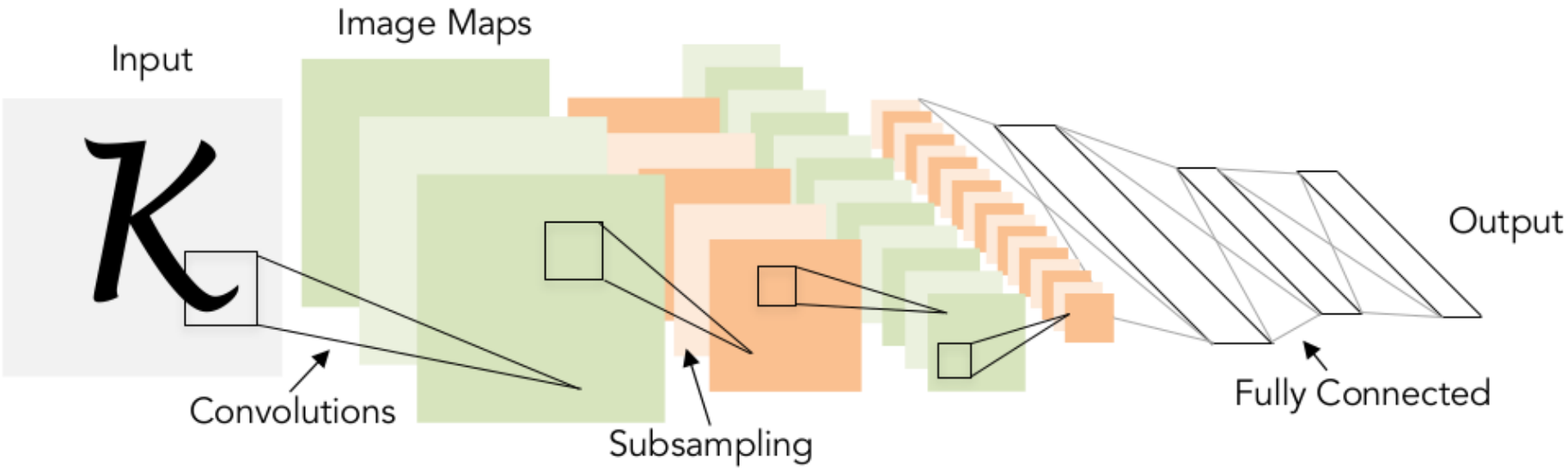


Image classification

CNN were not invented overnight.
1998 LeCun et al



# of transistors

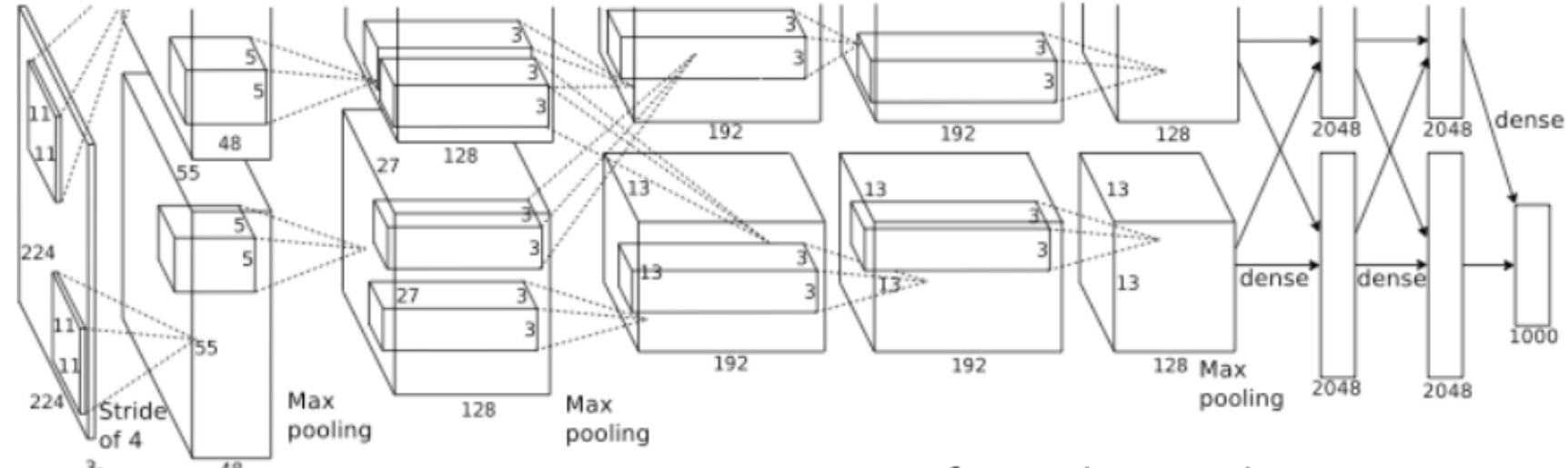
10^6

# of pixels used in training

10^7 NIST

Image classification

2012 Krizhevsky et al.



# of transistors

10^9

GPUs

# of pixels used in training

10^{14} IMAGENET

Why ConvNets

- Challenges of computer vision problem: Inputs can be really big.
- E.g for an image with 1000 x 1000 x 3 pixels (3 channels RGB)
  - We have about **3m** input features
  - If we use fully connected network and say we have 1000 hidden units then we get **3b** parameters which is very very large.
  - To train a NN with billion parameters is a bit infeasible.

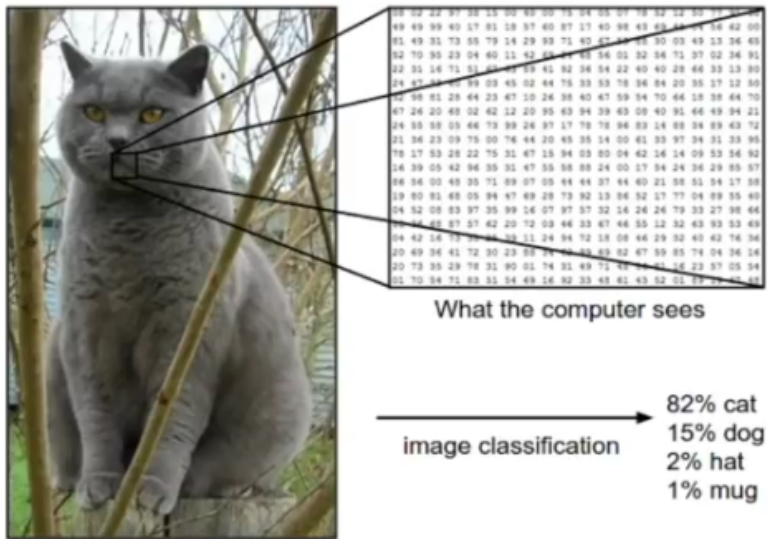


Figure: How Humans and Computers See an Image

Idea : Convolution (Parameter sharing) !

Convolution Process

- The Convolution operation is one of the building blocks of Convnets.
- Useful to detect edges.

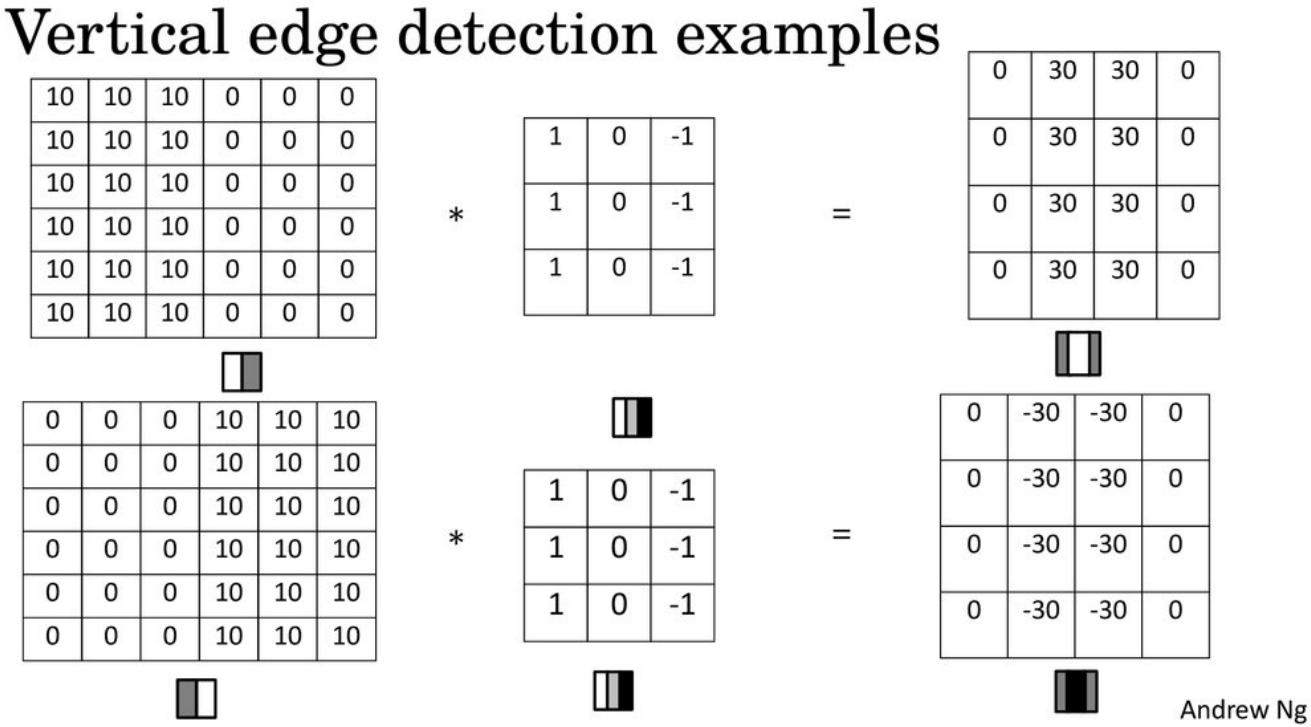


Figure: Edge detection for images [Andrew Ng]

Convolution Process

- To detect edges in a complicated images parameters (filters) are learned using back propagation algorithm
- learn low level features from data

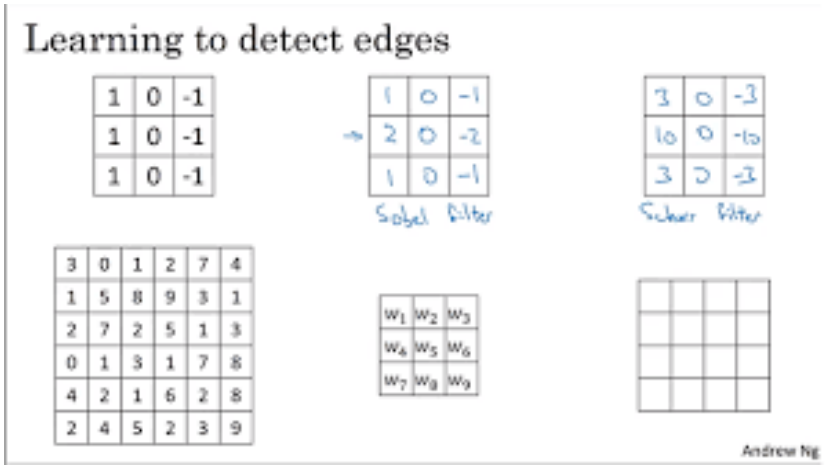


Figure: filter values are to be learned

Padding

- $n \times n$ image convolved with $f \times f$ filter produce $n - f + 1 \times n - f + 1$ dimension output image.
  - Shrinking of image outputs
  - Through away of information from the edges images
- Solution: Zero Pad the image before applying convolution to preserve spatial dimension.

0	0	0	0	0	0	0	0
0	3	1	1	2	8	4	0
0	1	0	7	3	2	6	0
0	2	3	5	1	1	3	0
0	1	4	1	2	6	5	0
0	3	2	1	3	7	2	0
0	9	2	6	2	5	1	0
0	0	0	0	0	0	0	0

Figure: Padding

Valid and Same Convolution

- Two common choices of Padding
- If we have $n \times n$ image and convolve with $f \times f$ filter with padding amount P then dimension of the output is $n + 2p - f + 1 \times n + 2p - f + 1$
  - Valid convolution: No padding $(P = 0)$
  - Same convolution: Pad so that the output size is same as the input size $\left(P = \frac{f-1}{2}\right)$

Strided Convolution

Stride : by how match is the filter shifted ?

- $n \times n$ image and $f \times f$ filter is convolved with padding p and stride s to give:

$$\left[\frac{n+2p-f}{s}+1 \times \frac{n+2p-f}{s}+1\right]$$

Pooling

- Makes the representation smaller and manageable.
- Operates over each activation maps.
- Max and average pooling

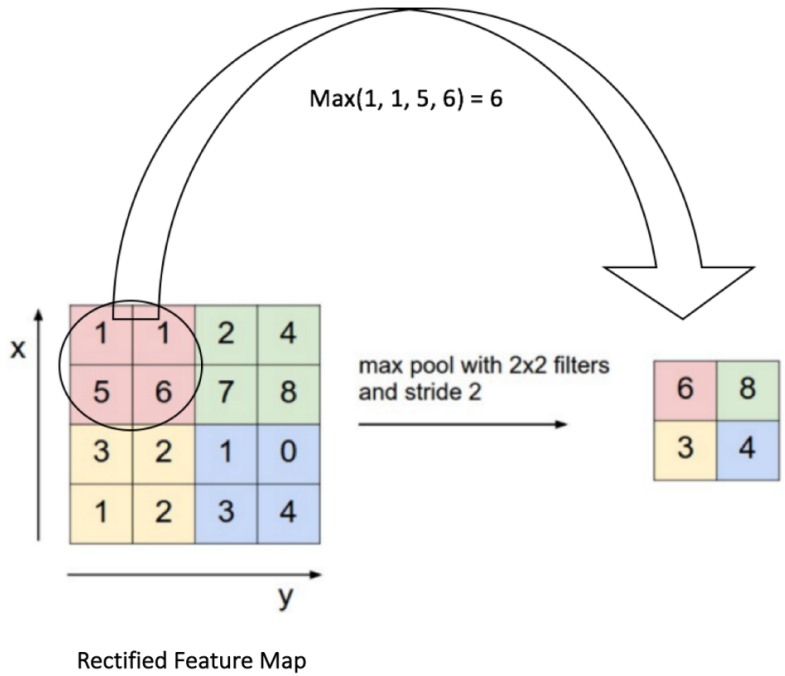


Figure: Maxpooling

- No parameters to be learned

Fully connected

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks.
- flatten(stretch) in to **1D** vector
- classifier.

ConvNets hyperparameters

Requires four hyperparameters

- The number of filters **K**
- The spatial extent **f**
- The stride **s**
- The amount of zero padding **p**

With parameter sharing they introduces $f \times f \times D_1$ weights per filter ,for a total of $(f \times f \times D_1) \times K$

layer summary

- Convolutional layers(CONV)
- Pooling layers(POOL)
- FullyConnected(FC)
  - stack them

$$Input \rightarrow CONV \rightarrow POOl \rightarrow CONV \rightarrow POOl \rightarrow FC \rightarrow FC \rightarrow Softmax$$

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	208
POOL1	(14,14,8)	1,568	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,001
FC4	(84,1)	84	10,081
Softmax	(10,1)	10	841

Figure: ConvNet example

Putting all Together

Training set $\left(x^{(1)}, y^{(1)}\right) \ldots \left(x^{(m)}, y^{(m)}\right)$

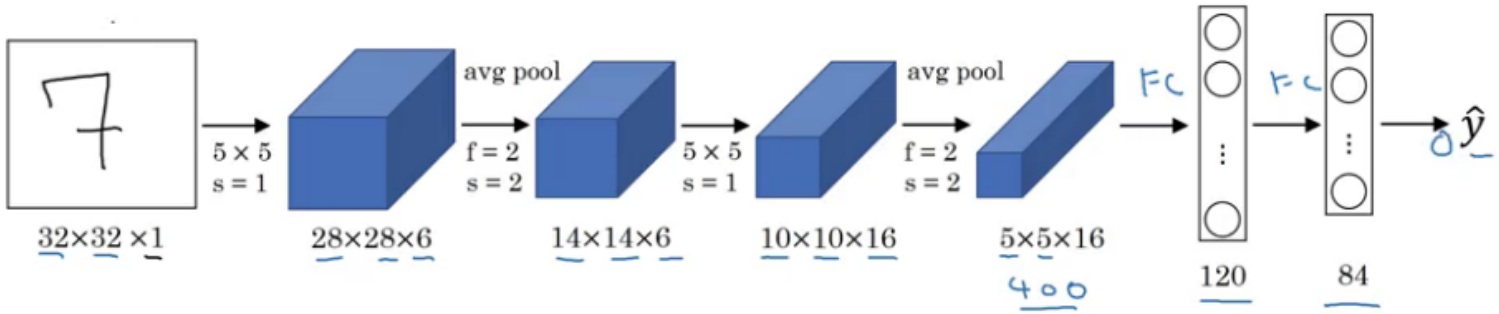


Figure: Digit recognizer architecture

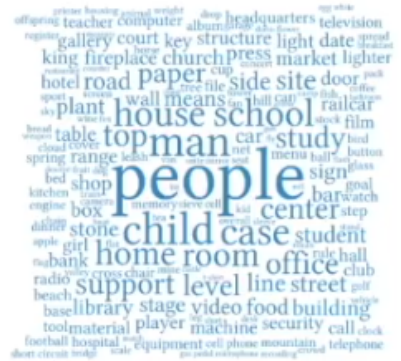
$$Cost = \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

Use Optimization Algorithm to optimize the parameters to reduce cost.

Famous Datasets

- MNIST :60K image data with 10 category
- ImageNet: 14+ iamges with 100 categories
- CIFAR-10(0): ImageNet simplified categories

MNIST: handwritten digits



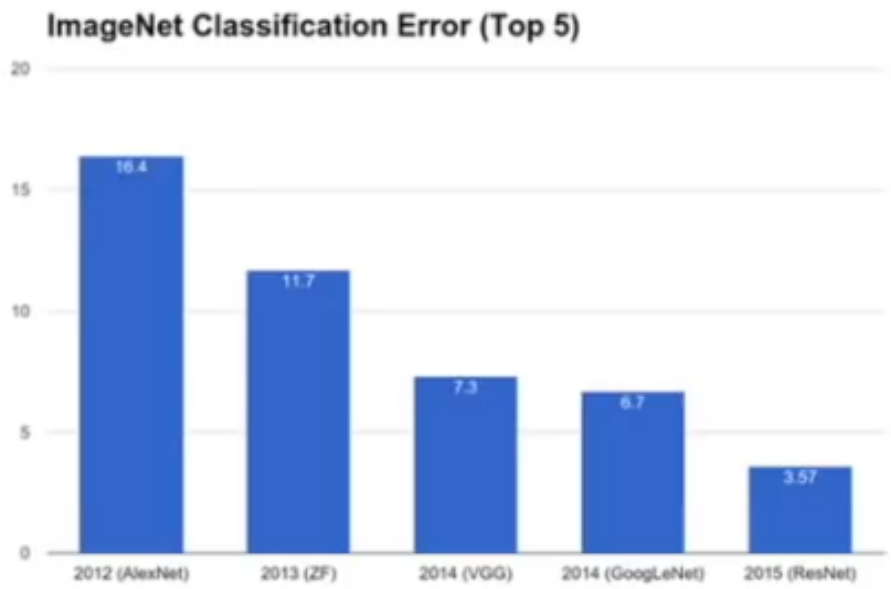
ImageNet: WordNet hierarchy

CIFAR-10(0): tiny images

Places: natural scenes

Classic ConvNets

Dataset→dataset of 14+million images
Competition → ILSVRC Imagenet large scale visual recognition challenge.



- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance

Application of classic ConvNets to OCR

- Realworld problems often have multiple classes:e.g OCR

When applied to recognize characters and digits

- Dataset $\rightarrow$ Dataset(MNIST) Solved problem .
- Latin, Arabic character recognizer solved Problem!

Previous works of Geez OCR

- Several researches has been done on Geez OCR using different classification approaches.
- difficult to benchmark as they are working on their own data sets .

Researchers	Dataset size	classfier	Accuracy(100%)
Dereje T.	5172-character images	ANN	61%
Million M.	7680-character images	SVM	90.37%
Yaragal A.	1010-character images	ANN	73.18%

Table: Previous results of Geez OCR

Dataset Preparation

- By generating characters in a image format in automatic way form Geez Unicode character in different fonts
  - we generate geez character images in 12 different fonts

link to datasets generated from geez multi fonts

Dataset Preparation

From images that contain Geez text by applying OCRpy tool.

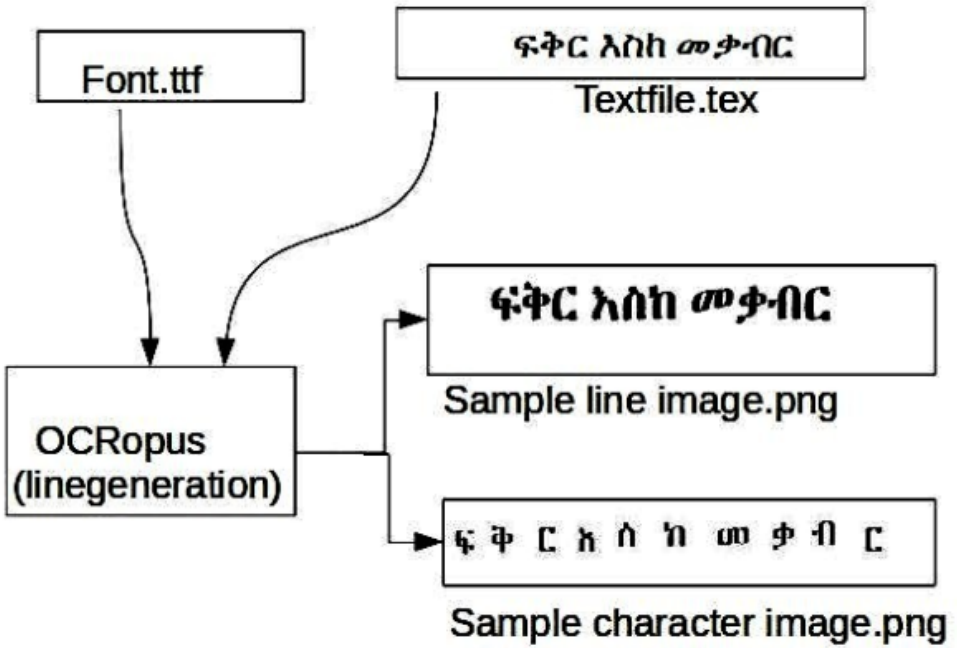


Figure: Pipline of Ocrpy To generate dataset

link Datasets prepared using OCrpy

Data augmentation

In Deep learning it is not who has the best algorithm that wins, it is who has the most data. To increase data sets different data augmentation techniques are applied

- Alpha blending
- Adding noise

Data set information

3751 Geez character datasets with $32 \times 32\times$ are prepared.

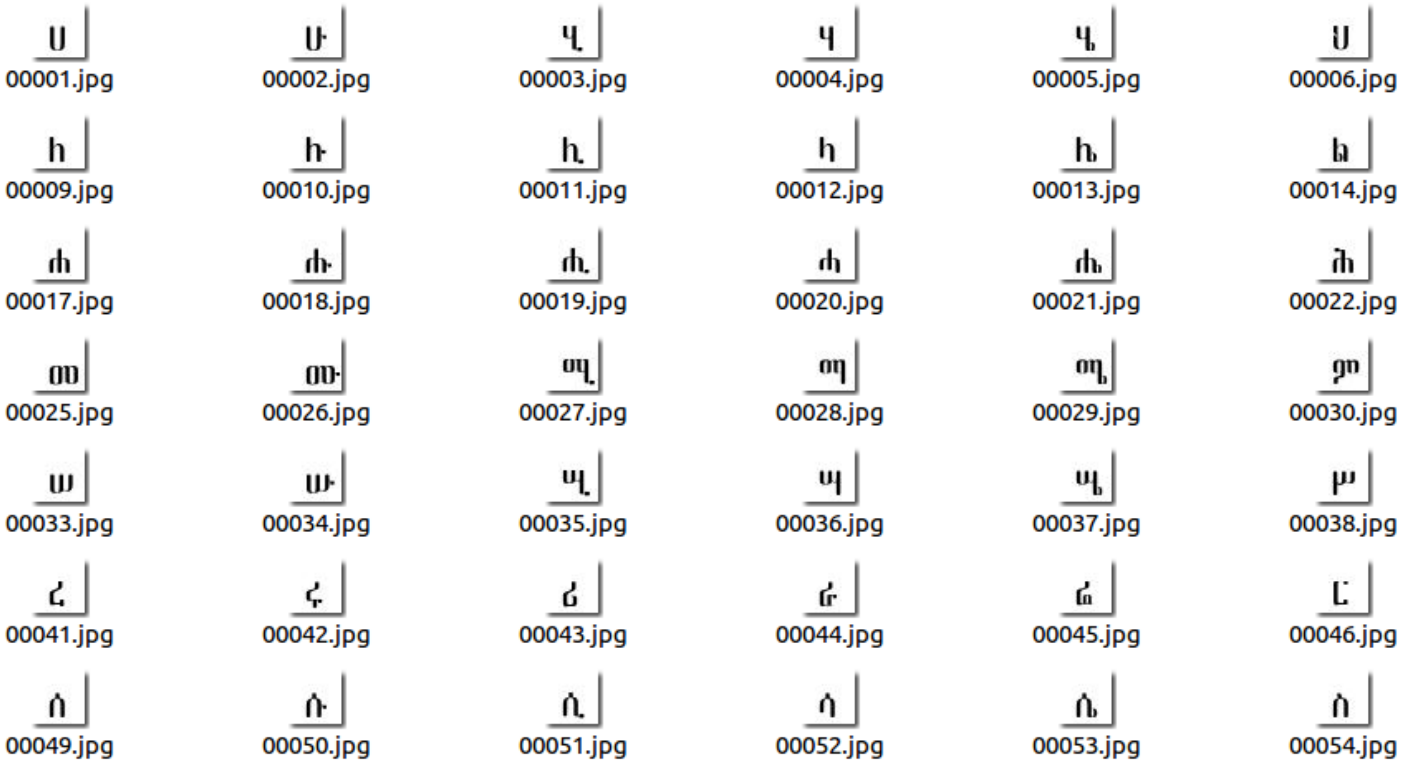


Figure: 42 sample datasets

Data labeling

All the 3751 Geez character datasets are labeled in to 37 family classes and 313 character classes.

Images	character class	family class
0001	0	0
0002	1	0
0003	2	0
0004	3	0
0005	4	0
0006	5	0
0007	6	0
0008	7	0
0009	8	1
.	.	.
.	.	.
.	.	.
3751	313	37

Table: labeled data

load Dataset and labels

- We use PyTorch DataLoader utilities.
- performs load, shuffle, normalize data and arrange it in batches.
- split them to training and testing with 4:1 ratio

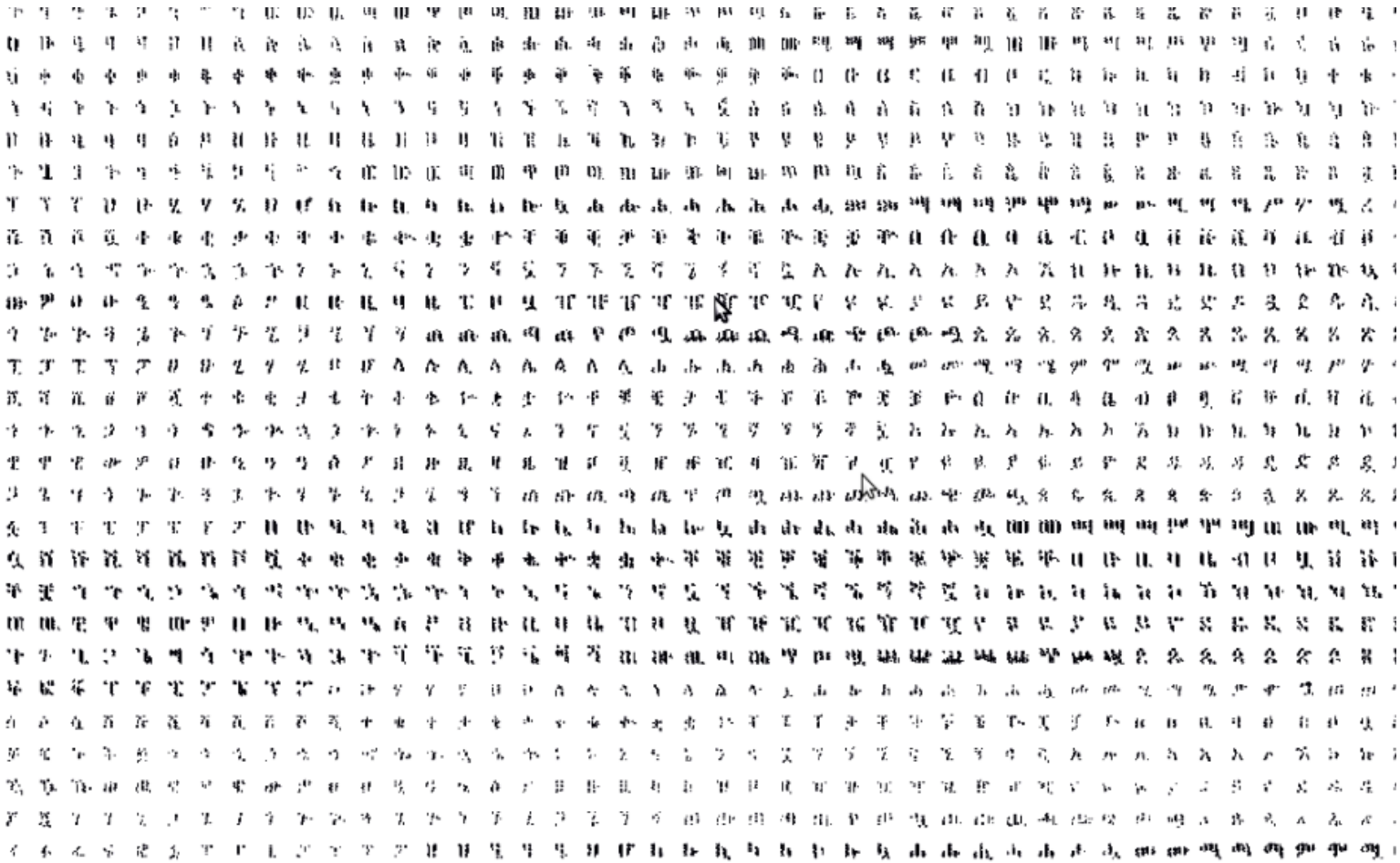


Figure: loaded datasets

OCR MOdel

```
ocr@OCR-user:~/Desktop/OCR conv net$ python conv.py
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 32, 32]             832
              ReLU-2         [-1, 32, 32, 32]               0
         MaxPool2d-3         [-1, 32, 16, 16]               0
            Conv2d-4         [-1, 64, 16, 16]          18,496
              ReLU-5         [-1, 64, 16, 16]               0
         MaxPool2d-6           [-1, 64, 8, 8]               0
            Conv2d-7          [-1, 128, 8, 8]          73,856
              ReLU-8          [-1, 128, 8, 8]               0
         MaxPool2d-9          [-1, 128, 4, 4]               0
           Conv2d-10          [-1, 256, 4, 4]         295,168
             ReLU-11          [-1, 256, 4, 4]               0
           Linear-12               [-1, 1000]       4,097,000
             ReLU-13               [-1, 1000]               0
           Linear-14                [-1, 500]         500,500
             ReLU-15                [-1, 500]               0
           Linear-16                 [-1, 37]          18,537
       LogSoftmax-17                 [-1, 37]               0
================================================================
```

Figure: Geez OCR model architecture

OCR MOdel parameters

Total params: 5,004,389
Trainable params: 5,004,389
Non-trainable params: 0

Layer Dimensions our Geez OCR MOdel

```
net.0.weight                torch.Size([32, 1, 5, 5])
net.0.bias                  torch.Size([32])
net.3.weight                torch.Size([64, 32, 3, 3])
net.3.bias                  torch.Size([64])
net.6.weight                torch.Size([128, 64, 3, 3])
net.6.bias                  torch.Size([128])
net.9.weight                torch.Size([256, 128, 3, 3])
net.9.bias                  torch.Size([256])
fc.0.weight                 torch.Size([1000, 4096])
fc.0.bias                   torch.Size([1000])
fc.2.weight                 torch.Size([500, 1000])
fc.2.bias                   torch.Size([500])
fc.4.weight                 torch.Size([37, 500])
fc.4.bias                   torch.Size([37])
ocr@OCR-user:~/Desktop/OCR conv net$
```

Figure: Geez OCR model architecture

Training Geez OCR MOdel

- load the image data to the model
- Initialize the weights and bias randomly
- Fix the input and output.
- Forward pass the inputs. calculate the cost.
- compute the gradients and errors.
- Backprop and adjust the weights and bias accordingly

Optimization and Over fit prevention

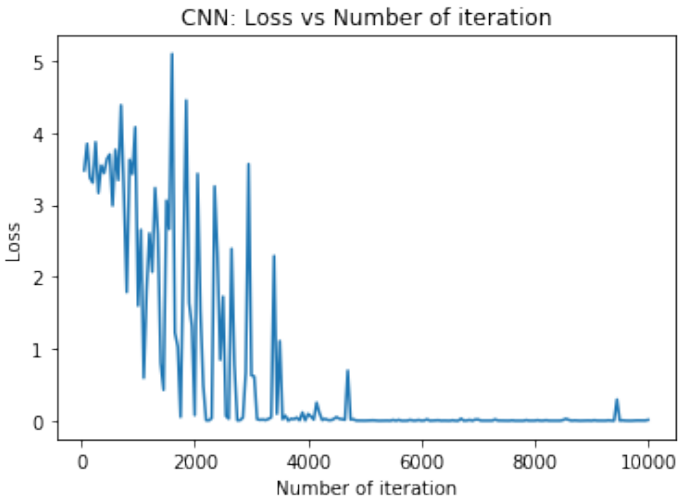
- Drop out(0.5)
- optimizer: SGD

Objectives of experiments

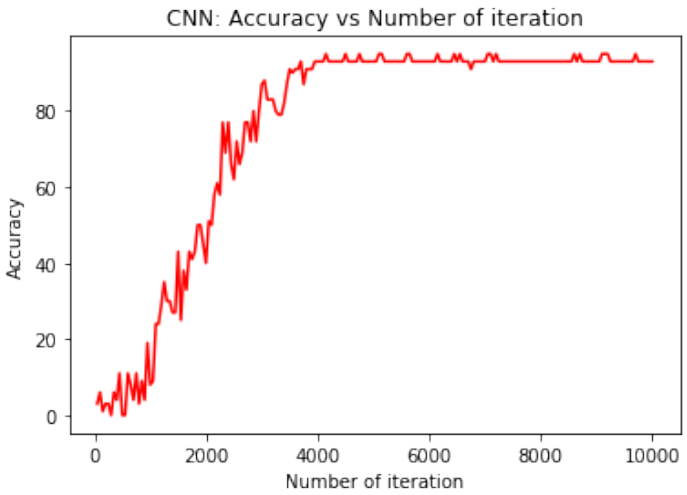
To quantify and visualize performance of our Model:

- To determine the best recognition accuracy and its corresponding iteration.
- visualize the connection weights and biases parameters
- To visualize learning progress
- visualize the connection weights and biases parameters and their gradients.
- Determine the best optimizer.

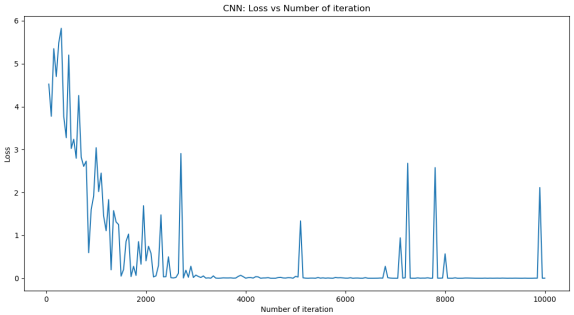
Results of the experiments



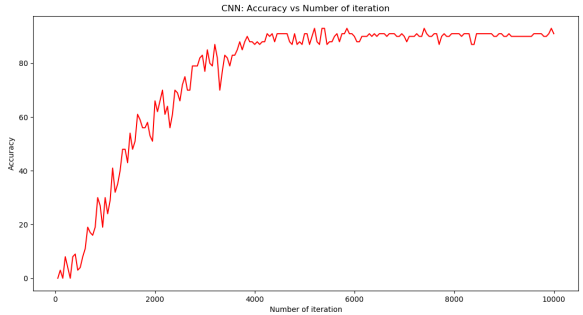
Family level Training loss



Family level Testing Accuracy



character level Training loss



character level Testing Accuracy

A representation of the terminal progressive bar for the family level training phase is as below:

```
3751
Iteration: 500  Loss: 3.700492620468139 6  Accuracy: 0 %
Iteration: 1000  Loss: 1.597695231437683  Accuracy: 8 %
Iteration: 1500  Loss: 3.059925794601440 4  Accuracy: 43 %
Iteration: 2000  Loss: 0.077308058738708 5  Accuracy: 40 %
Iteration: 2500  Loss: 1.724466562271118 2  Accuracy: 62 %
Iteration: 3000  Loss: 0.631373107433319 1  Accuracy: 87 %
Iteration: 3500  Loss: 1.108681082725525  Accuracy: 91 %
Iteration: 4000  Loss: 0.095301121473312 38  Accuracy: 93 %
Iteration: 4500  Loss: 0.051092091947793 96  Accuracy: 95 %
Iteration: 5000  Loss: 0.001446835813112557  Accuracy: 93 %
Iteration: 5500  Loss: 0.011312856338918209  Accuracy: 93 %
Iteration: 6000  Loss: 0.001089851488359272 5  Accuracy: 93 %
Iteration: 6500  Loss: 0.000235005441936664 28  Accuracy: 93 %
Iteration: 7000  Loss: 0.017861494794487953  Accuracy: 93 %
Iteration: 7500  Loss: 0.002349600661545992  Accuracy: 93 %
Iteration: 8000  Loss: 0.001798096462152898 3  Accuracy: 93 %
```

```
Iteration: 8500  Loss: 0.010310914367437363  Accuracy: 93 %
Iteration: 9000  Loss: 0.0004254833038430661  Accuracy: 93 %
Iteration: 9500  Loss: 0.020055299933999777  Accuracy: 93 %
Iteration: 10000  Loss: 0.012136055156588554  Accuracy: 93 %
```

A representation of the terminal progressive bar for the character level training phase is as below:

```
3751
Iteration: 500  Loss: 3.025829792022705  Accuracy: 4 %
Iteration: 1000  Loss: 3.0416088104248047  Accuracy: 30 %
Iteration: 1500  Loss: 0.04989071190357208  Accuracy: 54 %
Iteration: 2000  Loss: 0.4065481126308441  Accuracy: 66 %
Iteration: 2500  Loss: 0.013221027329564095  Accuracy: 66 %
Iteration: 3000  Loss: 0.07309813797473907  Accuracy: 77 %
Iteration: 3500  Loss: 0.002918415702879429  Accuracy: 79 %
Iteration: 4000  Loss: 0.000945672916714102  Accuracy: 87 %
Iteration: 4500  Loss: 0.0003134546568617224 7  Accuracy: 91 %
Iteration: 5000  Loss: 0.04391350597143173  Accuracy: 91 %
Iteration: 5500  Loss: 0.002313050674274564  Accuracy: 88 %
```

```
Iteration: 6000  Loss: 0.002709972904995084  Accuracy: 90 %
Iteration: 6500  Loss: 0.0001737029233481735  Accuracy: 91 %
Iteration: 7000  Loss: 0.0003093659470323473  Accuracy: 90 %
Iteration: 7500  Loss: 0.0016097567277029157  Accuracy: 90 %
Iteration: 8000  Loss: 0.568343102931976 3  Accuracy: 91 %
Iteration: 8500  Loss: 0.0032605384476482 87  Accuracy: 91 %
Iteration: 9000  Loss: 0.0013699972769245505  Accuracy: 90 %
Iteration: 9500  Loss: 0.0004426774103194475  Accuracy: 90 %
Iteration: 10000  Loss: 0.0021088765934109688  Accuracy: 91 %
```

Results of the experiments

Summary of Geez character		
Iteration	Training loss	Testing accuracy(%)
500	3.02583	4
2000	0.01322	66
3500	0.00292	79
5000	0.04392	91
6500	0.00017	91
8000	0.56834	91
9500	0.00044	90
10500	0.00046	91

Table: Summary of Training process in character level

Tensor Board visualization

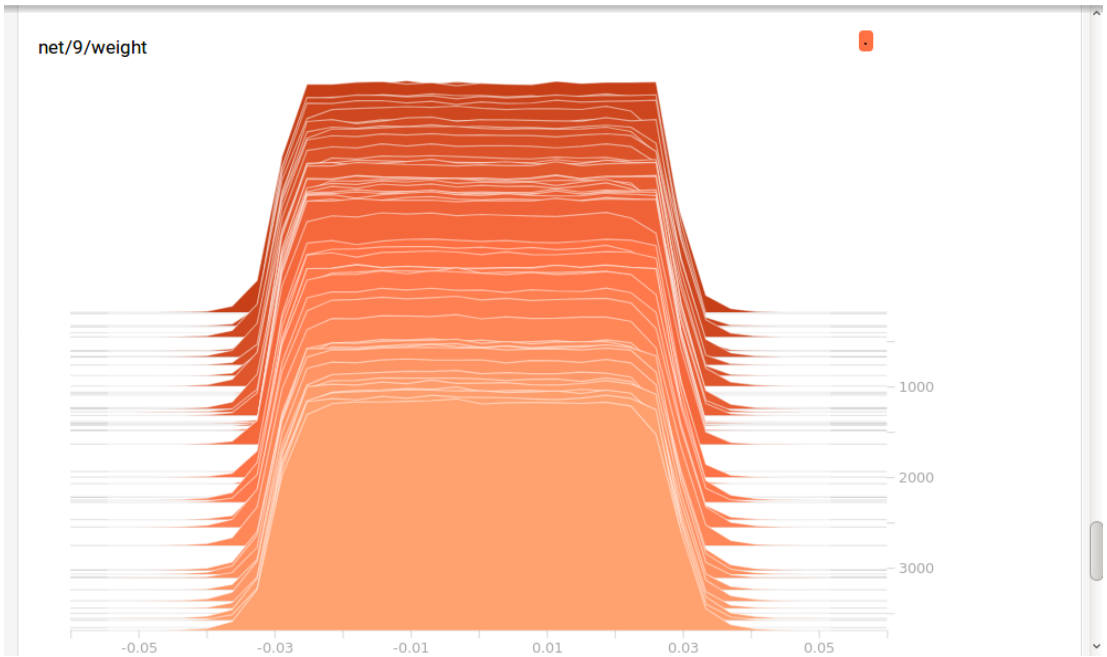


Figure: Histogram of weights for the last step (family level)

Tensor Board visualization

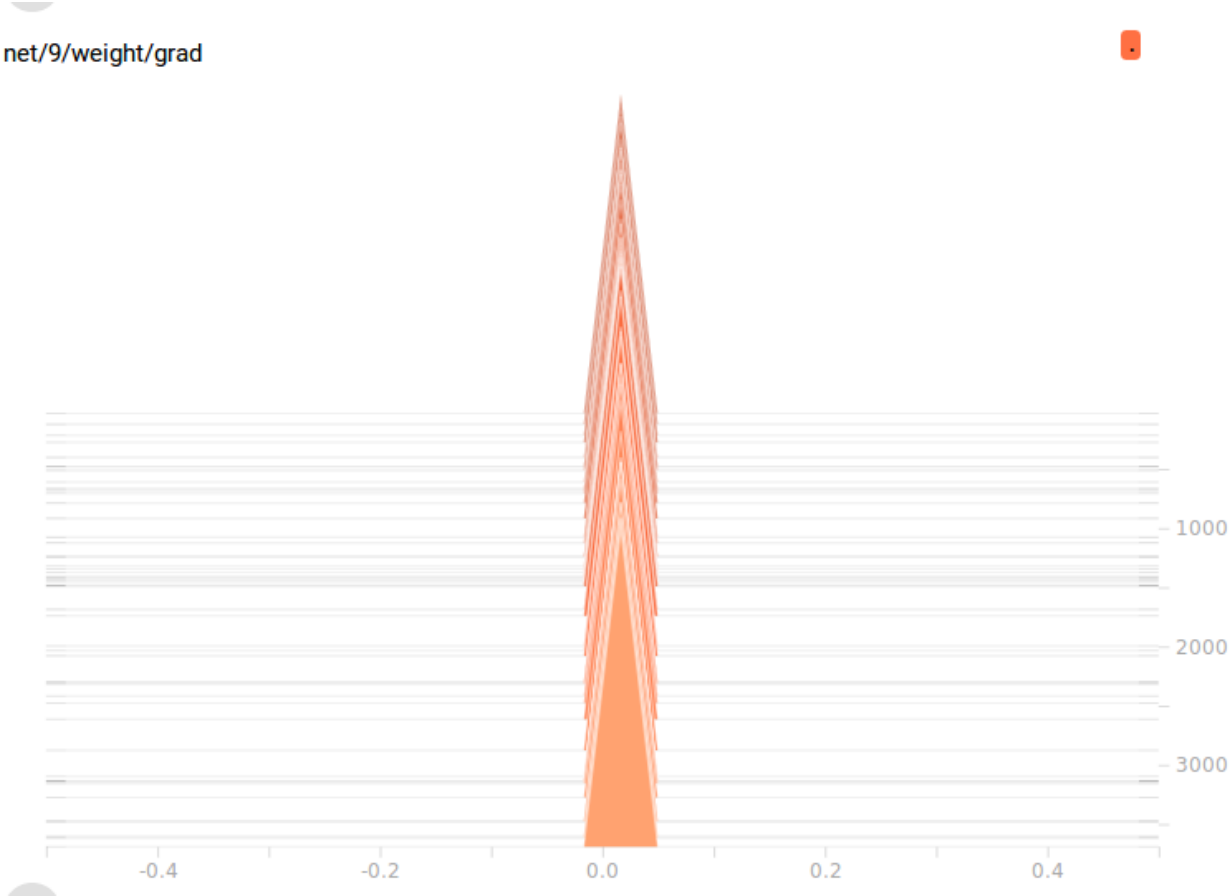


Figure: Histogram of gradient weights for the last step (family level)

Performance comparison

Researchers	Dataset size	classfier	Accuracy(100%)
Dereje T.[?]	5172-character images	ANN	61%
Million M.[?]	7680-character images	SVM	90.37%
Yaragal A.[?]	1010-character images	ANN	73.18%
Ours	3751-character images	DCNN	91%

Table: Performance comparison of related works.

Conclusions and future developments

- We experimentally developed a convolutional neural network model to recognize Geez character images.
- During training each family character exists 101 times on average.
- According to the experimental results maximum recognition accuracy is at 40 th epoch and 10500 th iteration.
- In this work we have also introduced a new data set preparation technique for future Geez character recognition.

- increase the number of datasets for a better result
- As an extended research, the OCR technology towards the recognition of Geez document image can be further referenced as word and sentence level, in parallel,large corpus of historical and printed real life document images will be prepared with a benchmark experimental result.

All references that I have used are in **Bibliography.bib** file.

Thanks for your attention