Facial Keypoint Detection

- SARTHAK AGARWAL

Problem

- The task was to detect facial key-points on greyscale images of faces.
- Facial key-points are centers and corners of the eyes, eyebrows, nose and mouth, among other facial features.

Tools and Infrastructure

- Keras 1.2.2
- Tensorflow 1.0.1
- AWS GPU g2.2xlarge
- CUDA 8.0
- Python 3.5.2

- Challenge: Finding an AMI on AWS which had all the above versions.
  - Solution: Updated all the packages in my own AMI

Dataset

- Total Images: 7049 with dimensions: 96x96x1
- Input (X) => Array of flattened images.
  - X is between 0 to 255 because greyscale images.
  - Scaled the input to float values between [0,1]
  - Shape of X for NN is (# of images, 9216)
  - Shape of X for CNN is (# of images, 96, 96, 1)

- Output(Y) => 30 (x, y) coordinates of the 15 facial keypoints.
  - Range of Y is from 0 to 96
  - Scaled the target o float values between [-1, 1]
  - Finally, shape of Y is (# of images, 30)

Baseline Models – 1 & 2

- Basic Neural Network with 1 hidden layer having 100 nodes with two sets of hyperparameters.
- Execution time = 3 mins on CPU with 100 epochs.
- Loss method = mean squared error

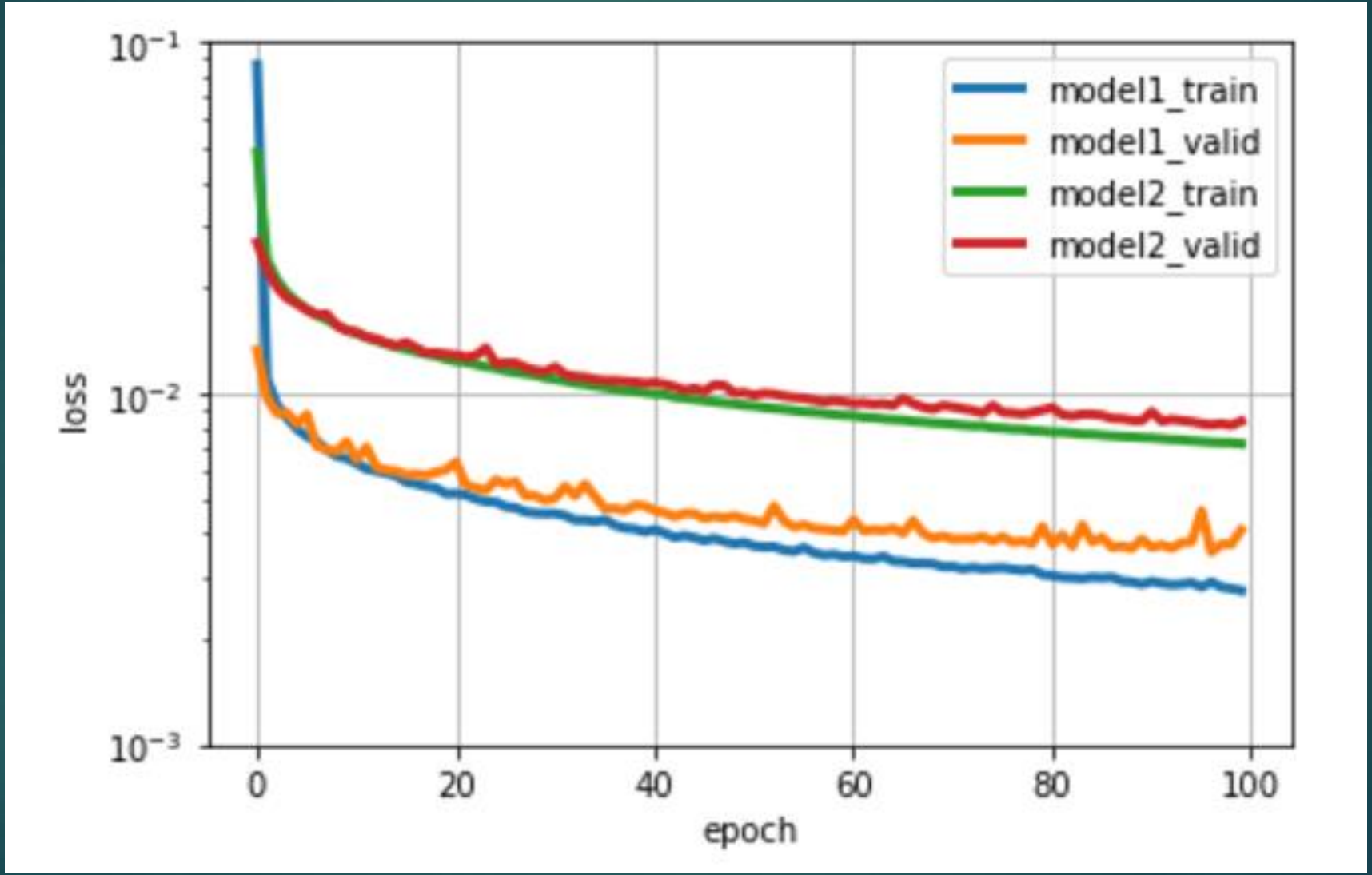
Parameter	Value
Learning rate	0.01
Momentum	0.9
Nesterov	True

Train Loss: 0.0024
Val Loss: 0.0037

Parameter	Value
Learning rate	0.01
Momentum	0.0
Nesterov	False

Train Loss: 0.0070
Val Loss: 0.0080

Performance comparison of the two baseline models



Model 3 - CNN

Layer	Name	Description	Outputs
0	Input	Input Image	1x96x96
1	Conv2d	32 filters of 3x3	32x94x94
2	Maxpool	Size 2x2	32x47x47
3	Con2d	64 filters of 2x2	64x46x46
4	Maxpool	Size 2x2	64x23x23
5	Conv2d	128 filters of 2x2	128x22x22
6	Maxpool	Size 2x2	128x11x11
7	Flatten	-	15488
8	Dense	-	500
9	Dense	-	500
10	Output	Output layer	30

- No of epochs = 500
- Time taken to run = 40mins
- Train Loss = 0.0012
- Val Loss = 0.0020

Models performance comparison

Model	Model Description	Epoch	Time	Train Loss	Val Loss
Model 1	NN with 1 hidden layer, 100 nodes	100	3mins CPU	0.0024	0.0037
Model 2	Model 1 + momentum=0, decay=0, nesterov=False	100	3mins CPU	0.0070	0.0080
Model 3	CNN	500	40mins GPU	0.0012	0.0020
Model 4	CNN – Flipped images	Yet to run	-	-	-

Prediction on Test images

Simple NN version 1

Simple NN version 2

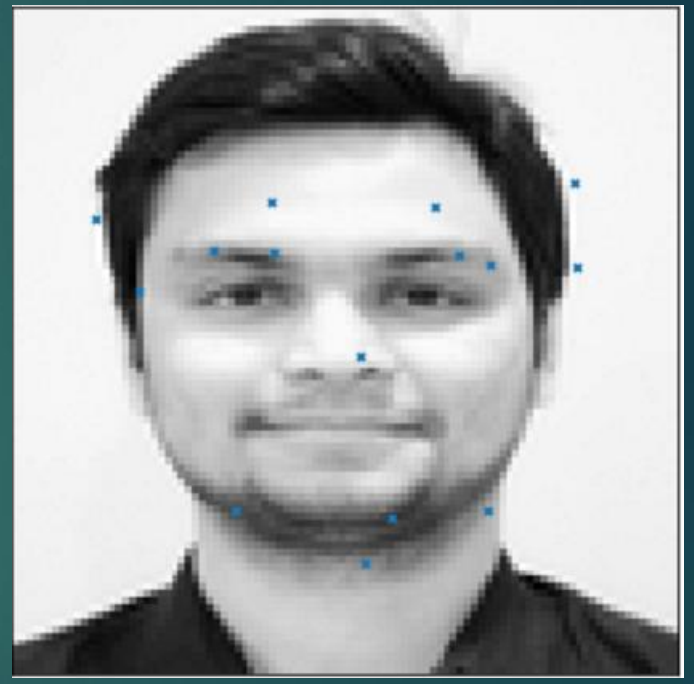
CNN

Prediction on new image

- For each model, I saved the model structure and weights for future prediction.

Simple NN version 1

Simple NN version2



CNN

Data Augmentation

- It is a process to artificially increase the number of training examples by applying transformations, adding noise etc.

- In my experiments:
  - flipped the images on vertical axis
  - Remapped the x and y coordinates for facial keypoints.

- Why did I do this?
  - The dataset where all the keypoints are present is 2100.
  - Therefore, to get better results I need more training data.

Future Work

- Run the CNN models for augmented data.
- Try different augmentation techniques.
- Run the models for 3 times more epochs.
- Run a deeper Neural network.

Thank you!