Cats vs Dogs Detector (CaDoD)
Phase 2

Group 19

Aishwarya Sinhasane, Himanshu Joshi

Sreelaxmi Chakkadath, Sumitha Vellinalur Thattai

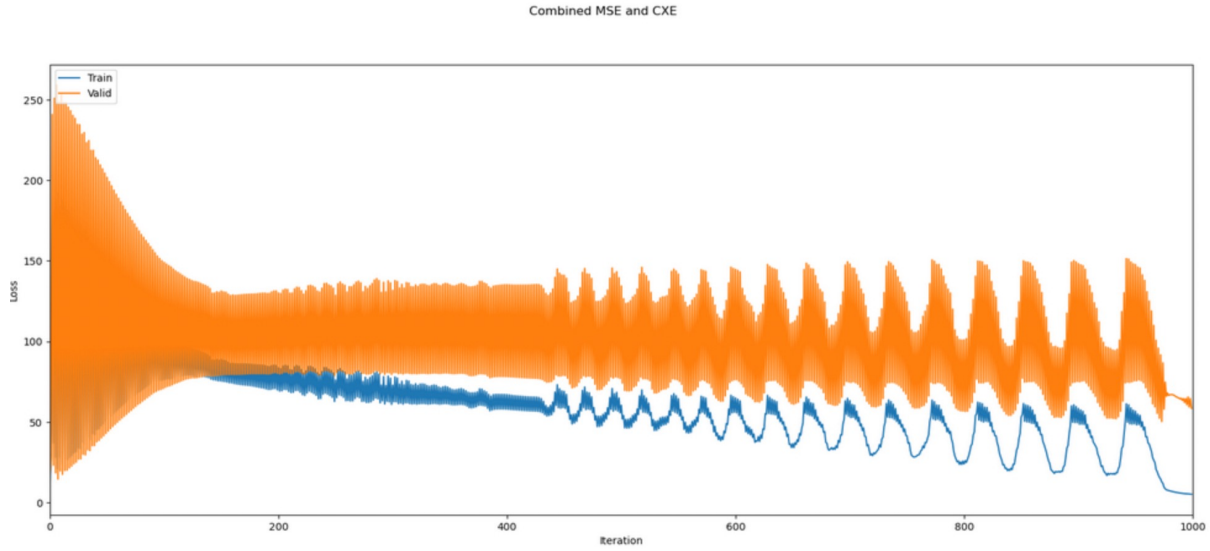
Agenda

- Past – Phase 1
- Present – Phase 2
  - Homegrown CXE + MSE Loss
  - Multi Layered Perceptron (MLP)
  - Multi headed MLP
  - Results and Discussion
- Conclusion & Next Steps

Past - Phase 1

- Performed EDA and rescaled the data set
- Built various models for image classification for boundary detection
- Baseline model:
  - Image classification - Gradient boosting
  - Boundary detection - Linear regression

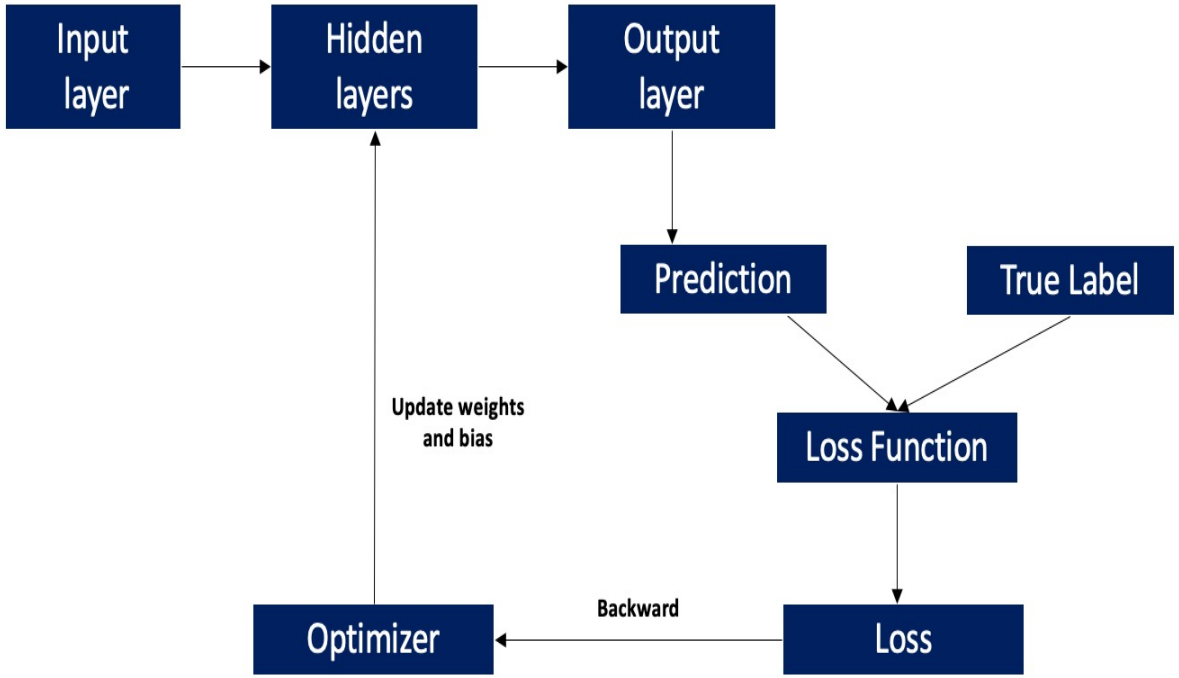
Phase 2 - Homegrown CXE + MSE



- **Homegrown Models**
  - We have implemented a logistic and linear regression from scratch
    - Elastic net (L1 and L2) was added to the homegrown model to decrease the MSE
  - Implemented functions to find log loss for classification and MSE for regression
- **Complex function**
  - We have defined a function which will combine the log loss of the logistic regression and MSE of the linear regression

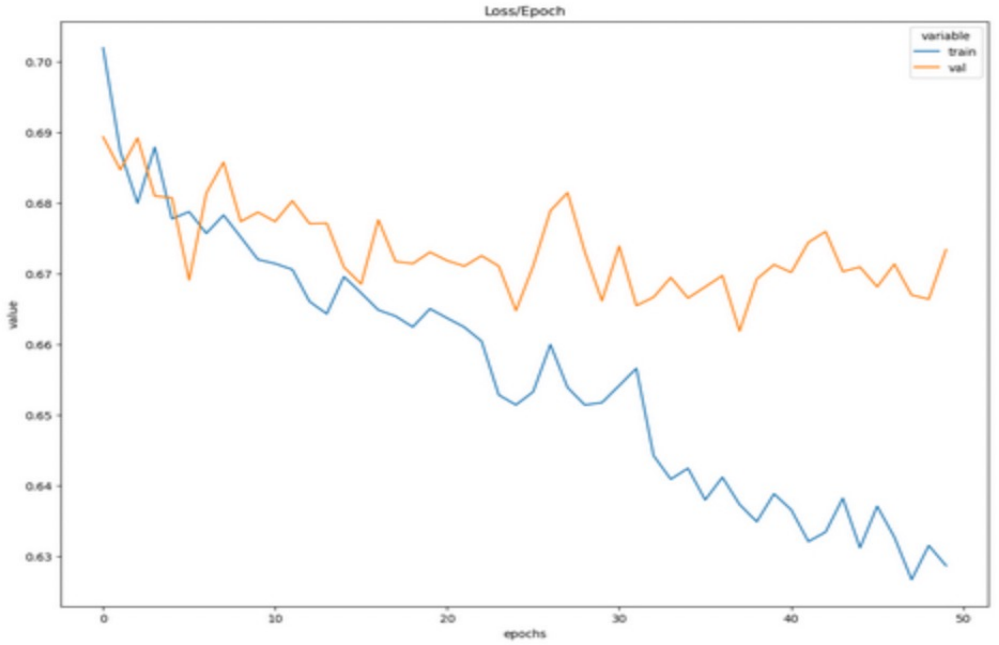
Multi-Layered Perceptron

- Built a neural network using sequential and OOP API in Pytorch
- Layers Used:
  - Input Layer
  - 2 hidden layers
    - Activation function – ReLU for classification and tanh for regression
    - Dropout layer
  - Output layer
- Loss functions used:
  - Cross Entropy – for classification; MSE – for regression
- Optimizer - Adam



Sequential Classifier For Image Detection

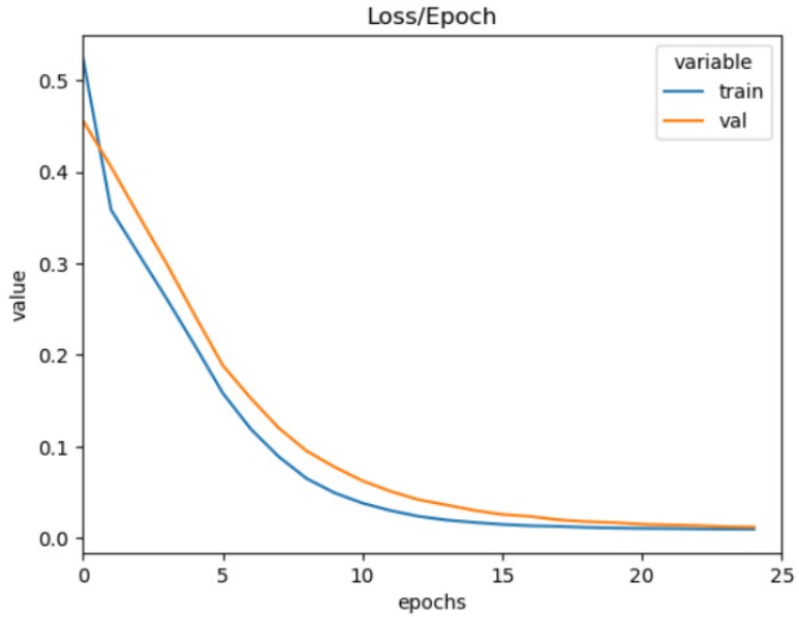
	exp_name	Train Acc	Valid Acc	Test Acc
0	Baseline Neural Network without Augmentation, ...	97.9	56.022	56.978
1	Neural Network without Augmentation and with R...	96.325	56.467	57.106
2	Neural Network without Regularization and Dropout	70.702	57.511	59.923
3	Neural Network without Dropout	69.702	58.556	57.875
4	Neural Network with Dropout (p = 0.2)	63.83	57.644	58.259
5	Neural Network with Dropout (p = 0.1)	66.83	59.089	59.283



- NN without augmentation overfits the data; training accuracy is very high, whereas the validation accuracy dropped to almost half
- Validation accuracy was improved by: Augmenting data, Adding dropout layer, and Regularization
- Accuracy and Loss graph for neural network with dropout layer (p = 0.1)
- **Optimizer – Adam; Loss function – Cross Entropy**

Sequential Regressor For Boundary Detection

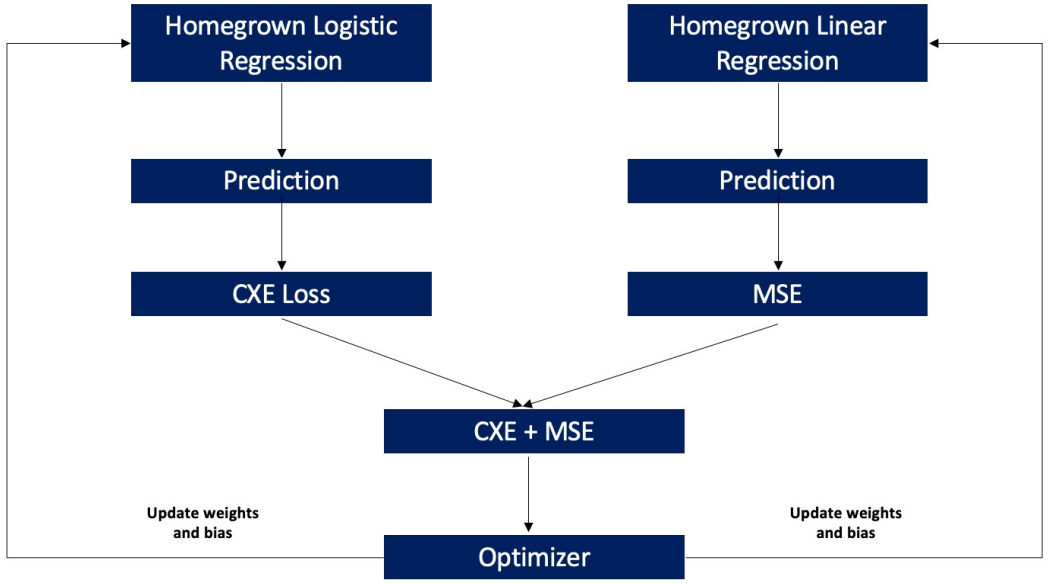
	exp_name	Train MSE	Valid MSE	Test MSE
2	Sequential Regression NN Dropout (p = 0.5)	0.009	0.011	0.011
0	Sequential Regression NN Dropout (p = 0.1)	0.009	0.011	0.011
1	Sequential Regression NN Dropout (p = 0.2)	0.01	0.012	0.012
3	Sequential Regression NN Dropout (p = 0.3)	0.01	0.013	0.013
4	Sequential Regression NN Dropout (p = 0.4)	0.01	0.012	0.012
5	Sequential Regression NN without dropout	0.01	0.012	0.012
6	Sequential Regression NN without dropout for w...	0.009	0.011	0.01
7	Sequential Regression NN without dropout for w...	0.01	0.022	0.022
8	Sequential Regression NN without dropout for w...	0.011	0.013	0.013

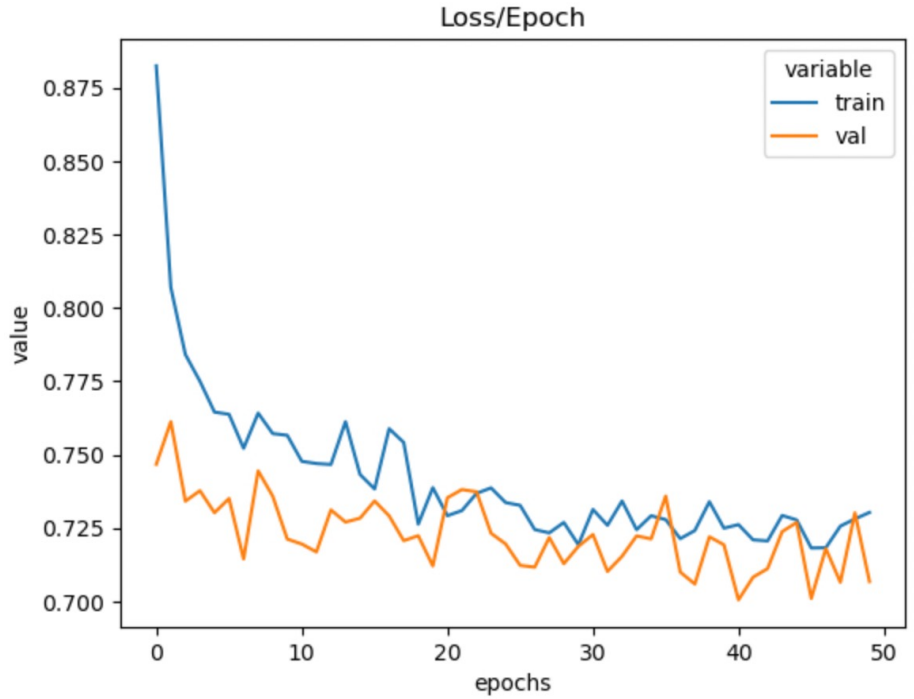


- Graph for neural network with dropout layer (p = 0.1)
- We see that loss is decreasing after every epoch for training and validation set
  - Also, there was not a huge difference in the MSE value after adding a drop out layer
- **Optimizer – Adam; Loss function - MSE**

Results in the next slide

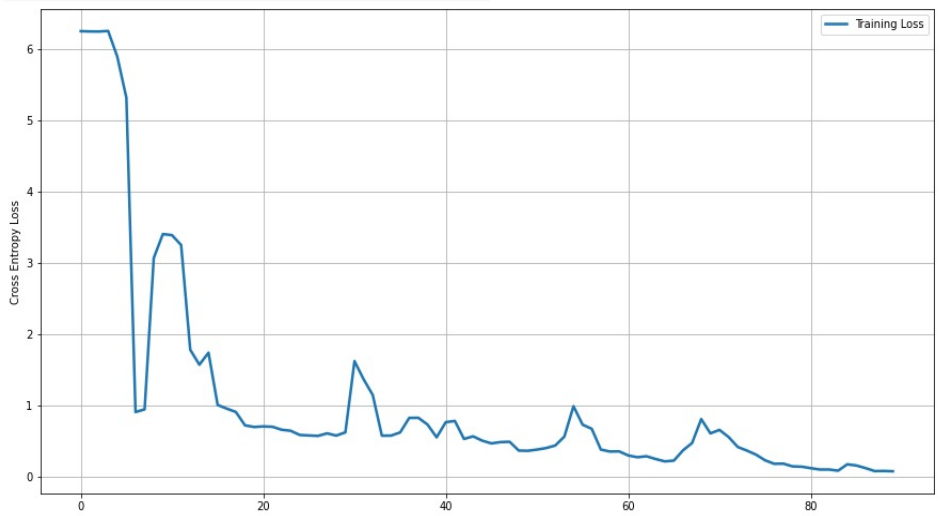
Complex Loss Function – CXE + MSE

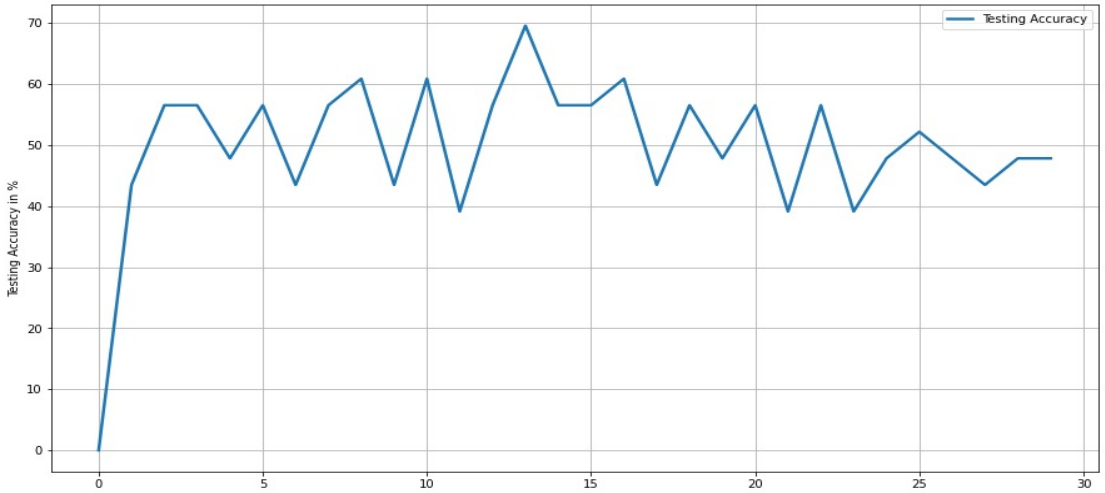




- CXE from classifier model and MSE from regressor model was combined and sent as a feedback to optimize the model
- We see that both training and validation loss are decreasing at each epoch

Convolution Neural Network





- Used a maxpooling layer to reduce the number of feature maps
- Used a softmax function in the output layer as an activation function
- Learning rate= 0.01, momentum=0.3
- Mini batch size = 30
- Optimizer - SGD; Loss function – Cross entropy

Results will be directional (and not very accurate) since it is based on only 2500 data points

Conclusion & Next Steps

- **Conclusion**
  - NN with augmentation, dropout layer and regularization helped us overcome overfitting and increase the accuracy to ~60%
    - Loss reduced at each epoch for both train and validation set

- **Next Steps**
  - Optimize the current CNN by experimenting with data augmentation, different optimizers, learning rates, and weight delays
  - Building a Transfer learning model using EfficientDet
  - Perform a t-test to compare the different models with this transfer learning model