Lohit Hostel

Automated Research paper Categorization

Problem Statement

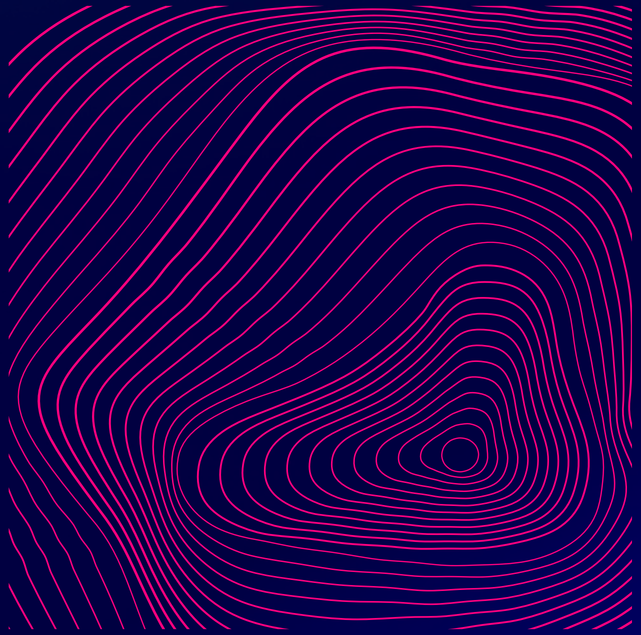
Building an Automated Research paper categorizer using Machine Learning and Deep Learning techniques.

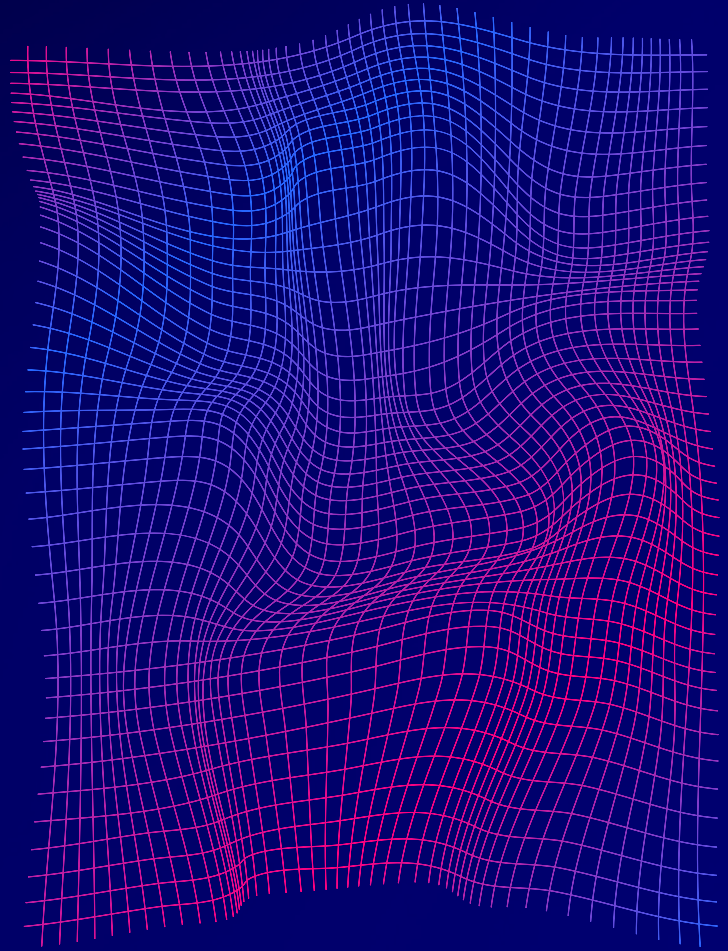
DATASET DESCRIPTION:

- The train and test datasets consist of multiple research papers.
- The features preset in the dataset are:
  - Title of research paper
  - Abstract of research paper
  - Categories to which the paper belongs to
  - Number of rows in train dataset is 51210
  - Number of rows in test dataset is 10974
  - Number of categories is 57

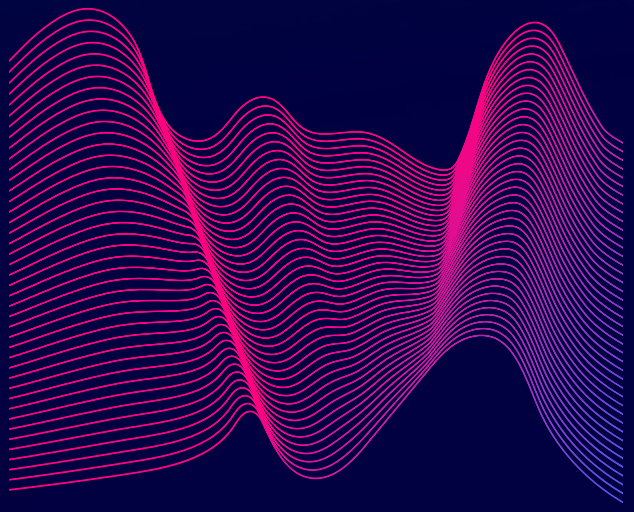
A peak at the dataset

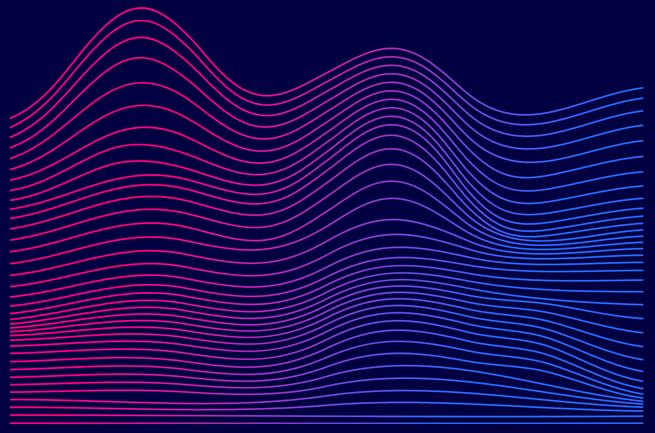
Title	Categories
Large deviations for Wishart processes	['math.PR']
Slicer Networks	['eess.IV', 'cs.AI', 'cs.CV']
New symmetry in nucleotide sequences	['q-bio.GN', 'q-bio.BM']
Modeling Credit Risk with Partial Information	['math.PR', 'q-fin.RM']
A Semantic Grid Oriented to E-Tourism	['cs.DC']

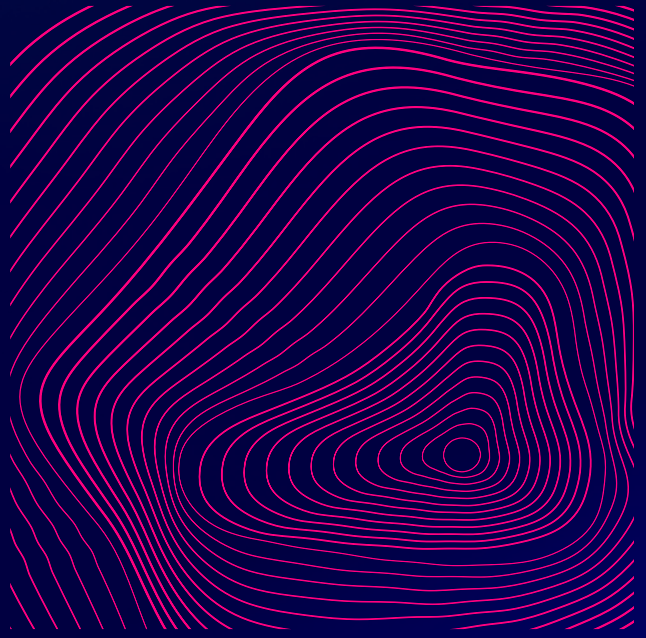


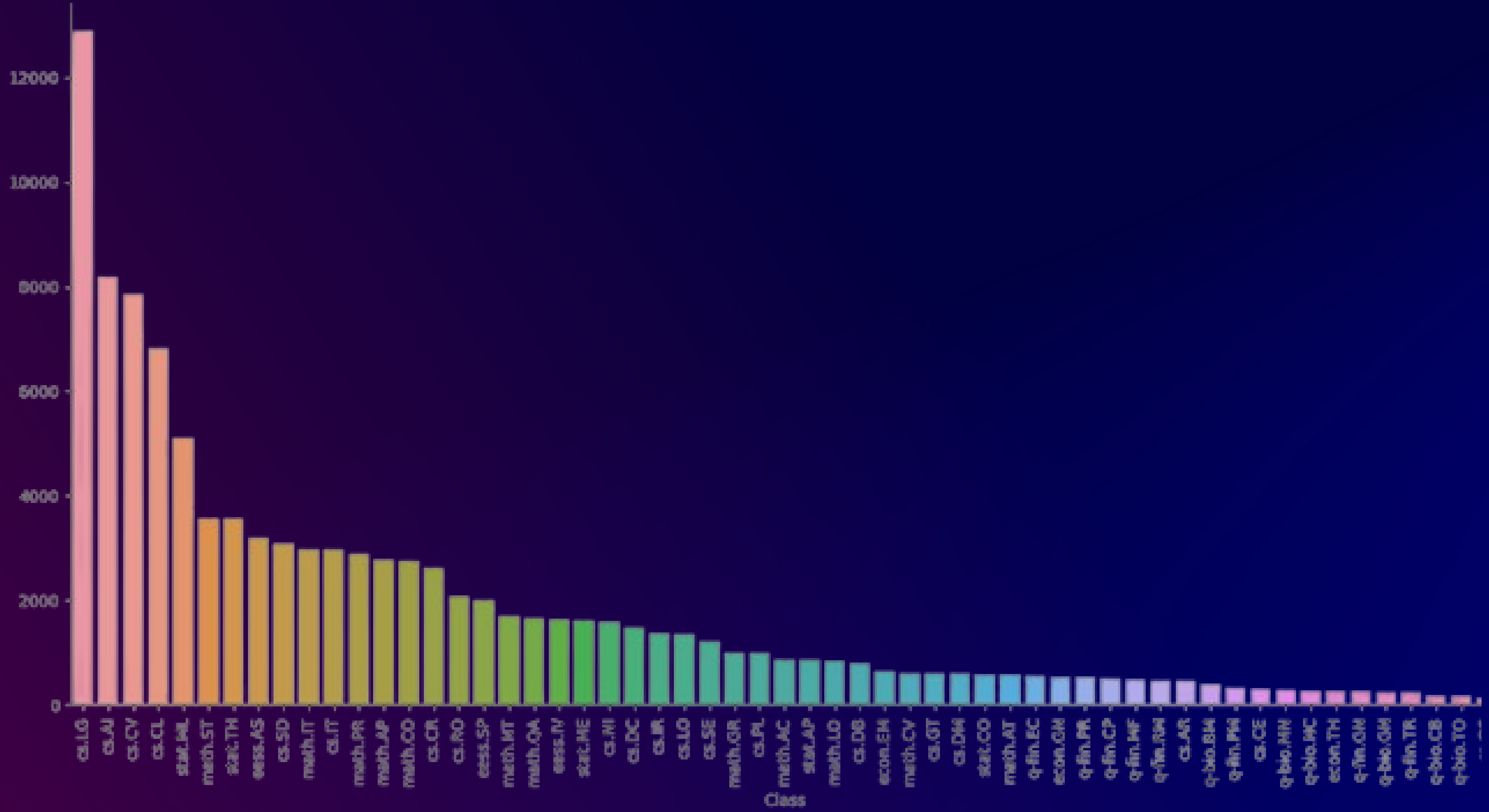


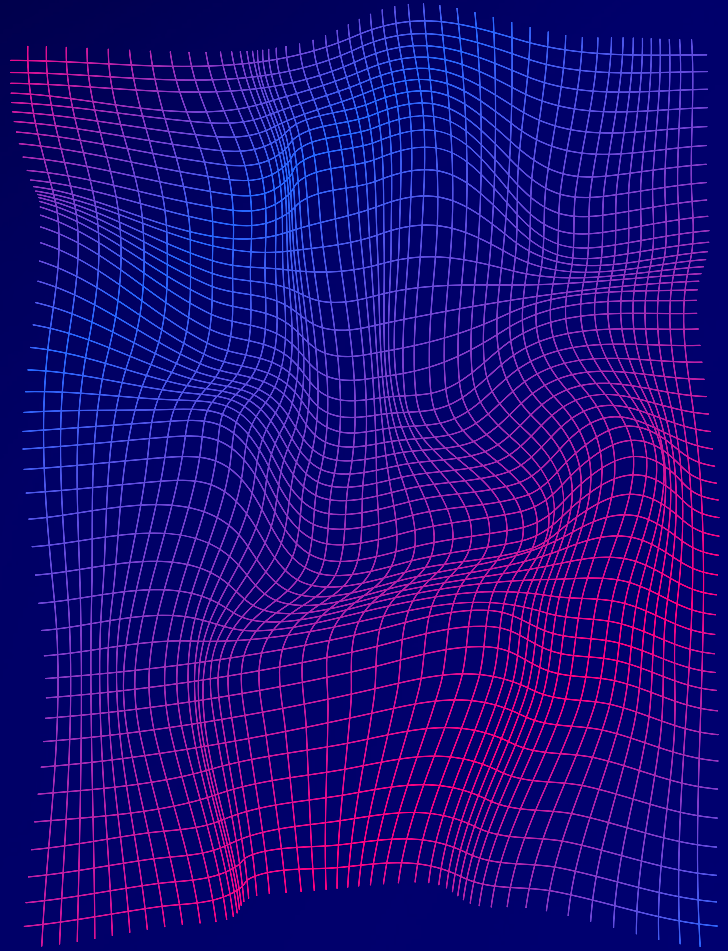
Class Imbalance











Data Preprocessing

- Class Weighting:
  - Assigned higher weights to minority classes during training to penalize misclassifications in these classes more heavily.
- Resampling:
  - Resampled the dataset to balance the class distribution. This involved randomly selecting samples from the majority class and creating synthetic samples for minority classes.

Text Preprocessing Summary

1 Merged the ‘Abstract’ and the ‘Title’ columns into a column called ‘Context’ which is finally used for Prediction.

2 This was followed by decontraction of some word like won’t ,can’t to ‘will not’ and ‘can not respectively’

3 We removed all the punctuation and stop word from the test.

4 Then we proceeded with stemming all the word to their root word . (Like ‘Happier’ to ‘Happy’ , ‘Programming’ to ‘Program’ and so on)

5 Then we formed a vocabulary using all the word present in the text.

6 Finally each row in the dataset was converted into a vector where each word in text was replaced with its position number in the vocab

Approaches

DL based approaches

CNN and Bi-LSTM

Gave a Public F1 score of 0.56

- Used an Embedding Layer
- Fed the embedding outputs to a 1D Conv layer
- Used a bidirectional LSTM layer followed by feeding it into a max pooling layer
- Feeded the outputs to subsequent dense layers
- Used an output layer with a sigmoid activation

Bert transformer model

Gave a Public F1 score of 0.61

- Used a pre-trained Bert model for multi-label classification
- Fine-tuned it on the train dataset

Dense Neural Network

Gave a Public F1 score of 0.65

- Used 3 dense layers with BatchNormalization and Dropout
- Used swish activation for initial layers with leaky relu for the last layer
- Used adam optimizer for model training

ML based approaches

XGBoost Classifier

Gave a Public F1 score of 0.54

- Used XGBoost with calibrated classifier CV
- Created a pipeline for each category using Tfidf Vectorizer and OnevsRestClassifier for multi-label classification

Support Vector machine

Gave a Public F1 score of 0.62

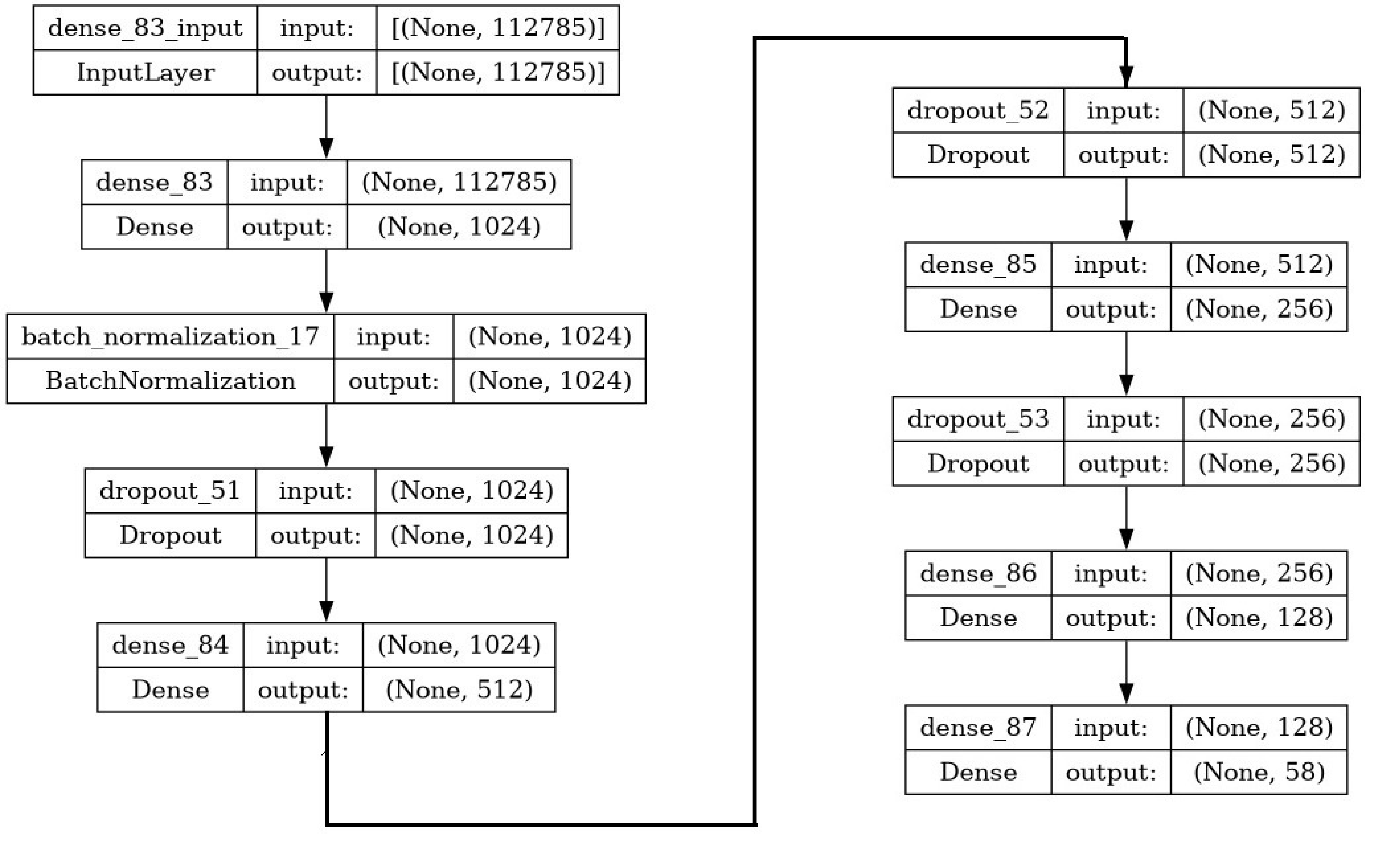
- Used SVM with calibrated classifier CV
- Created a pipeline for each category using Tfidf Vectorizer and OnevsRestClassifier for multi-label classification

Final approach

Dense Neural Network

- 4 densely connected layers with 1024, 512, 256 and 128 units.
- Swish activation function for the first three dense layers for its smooth, non-monotonic properties and leaky relu for the last dense layer and sigmoid activation for the output layer.
- Used regularization techniques like dropout for each of the three layers(0.6, 0.5 and 0.2) and BatchNorm for the first layer.
- Used initializers like HeNormal, HeUniform and GlorotUniform for the final layer to maintain the scale of gradients.
- Used Adam optimizer and binary crossentropy loss while training the model and tracking the performance using accuracy, precision and recall metrics.

Architecture



dense_83_input	input:	[(None, 112785)]
InputLayer	output:	[(None, 112785)]

dense_83	input:	(None, 112785)
Dense	output:	(None, 1024)

batch_normalization_17	input:	(None, 1024)
BatchNormalization	output:	(None, 1024)

dropout_51	input:	(None, 1024)
Dropout	output:	(None, 1024)

dense_84	input:	(None, 1024)
Dense	output:	(None, 512)

dropout_52	input:	(None, 512)
Dropout	output:	(None, 512)

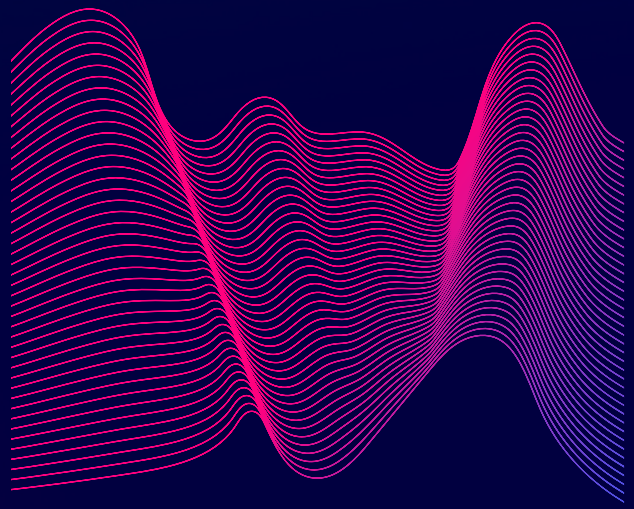
dense_85	input:	(None, 512)
Dense	output:	(None, 256)

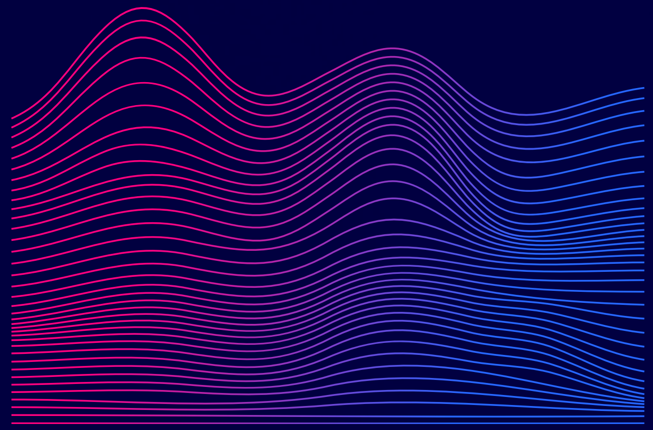
dropout_53	input:	(None, 256)
Dropout	output:	(None, 256)

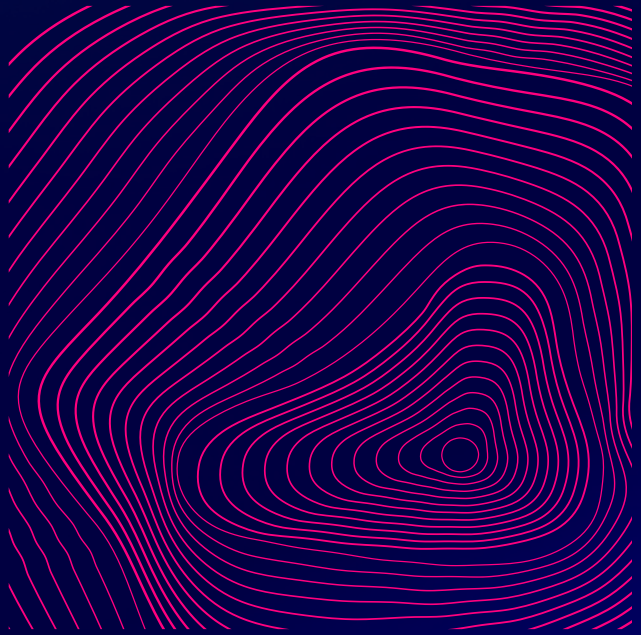
dense_86	input:	(None, 256)
Dense	output:	(None, 128)

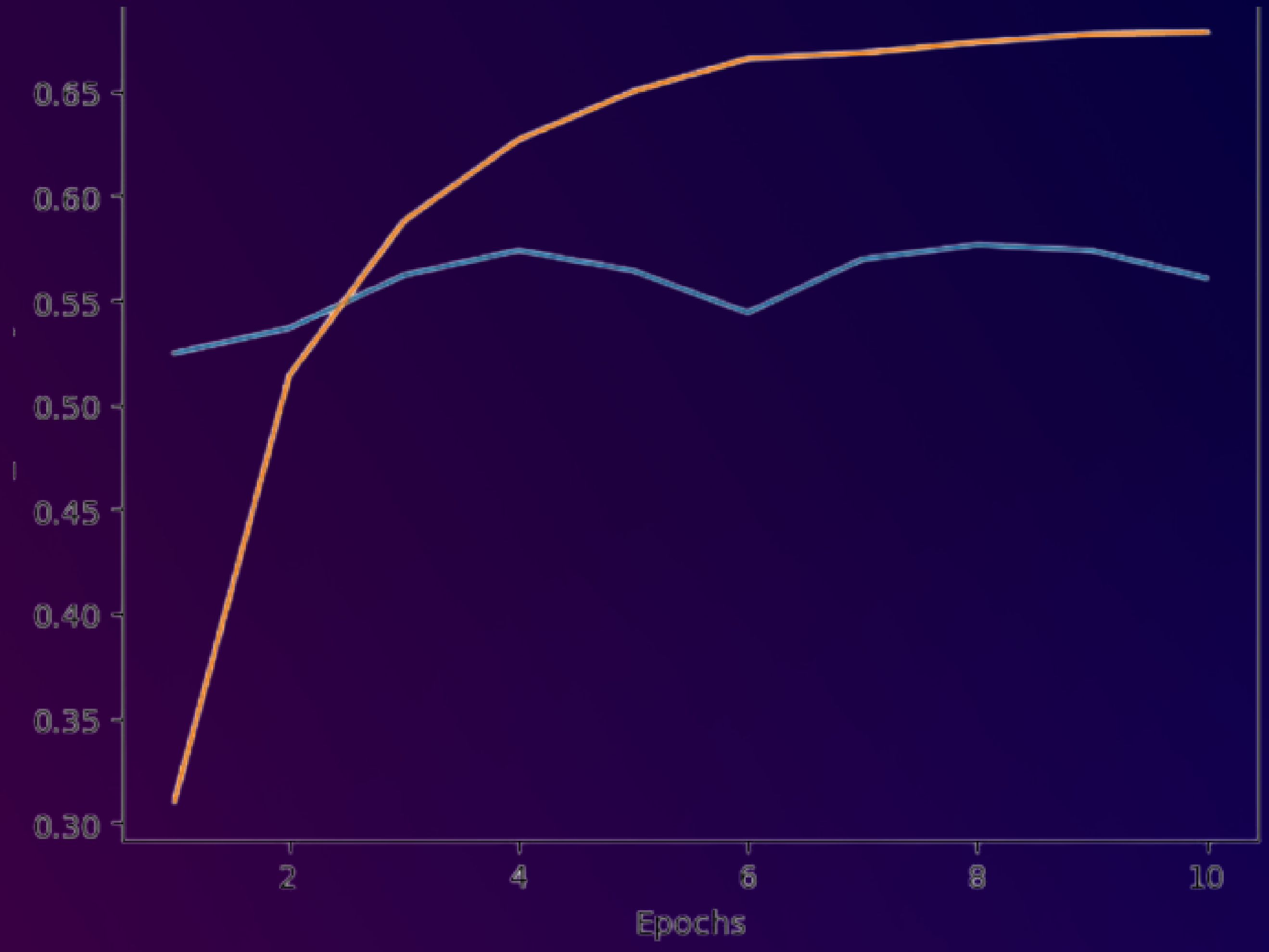
dense_87	input:	(None, 128)
Dense	output:	(None, 58)

Training Logs

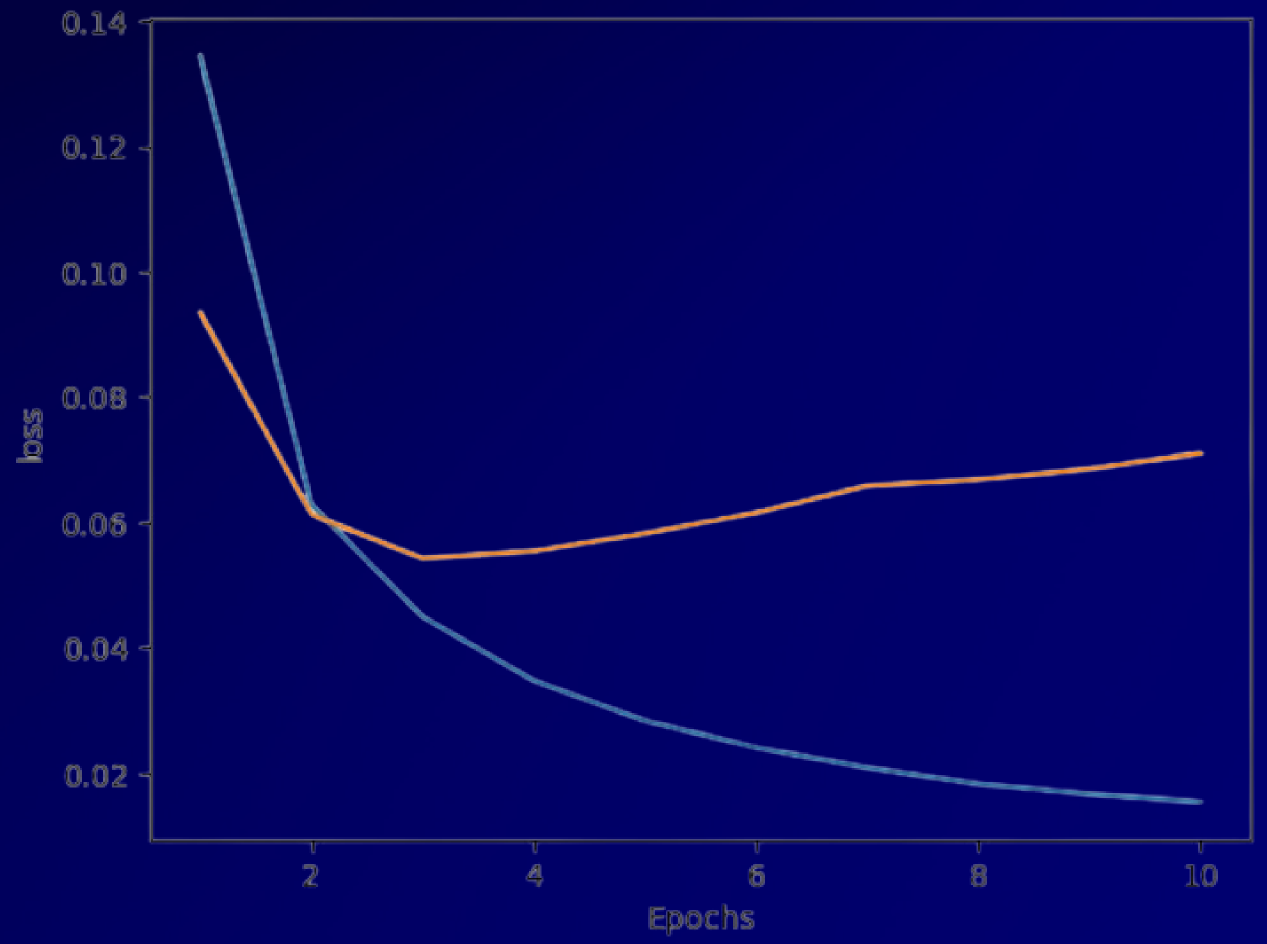








Training accuracy vs Validation Accuracy



Training loss vs Validation loss

Thank You