BT4012 In-Class Kaggle Competition Report

Author: Loh Hong Tak Edmund | A0199943H

Overview

This classification project aims to detect anomalies within data sampled from the Character Font Images Dataset. This project adopts a supervised learning approach in anomaly detection and employs multiple oversampling techniques, and machine and deep learning models to detect the anomaly. Subsequently, 2 of the best performing models were selected to be submitted on Kaggle.

Exploratory Data Analysis

The train.csv dataset loaded has 401 columns and 72139 rows. Among the 401 columns, the first 400 columns contain RGB pixel values (0-255) representing colours in the 20x20 square image. The last column indicates anomaly (0: Negative, 1: Positive).

The train.csv dataset is highly imbalanced, with 65581 negative-class rows and 6558 positive-class rows (figure 1). To gain clearer understanding of the dataset, the data was visualised as images, along with their corresponding labels (figure 2).

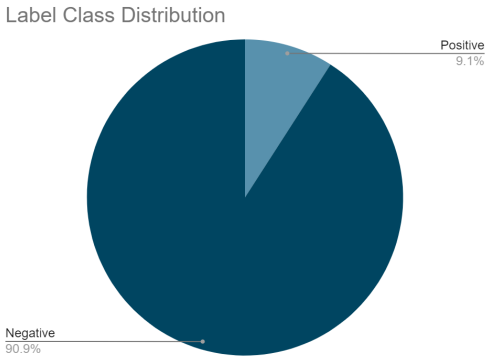

FIgure 1: Class Label Distribution

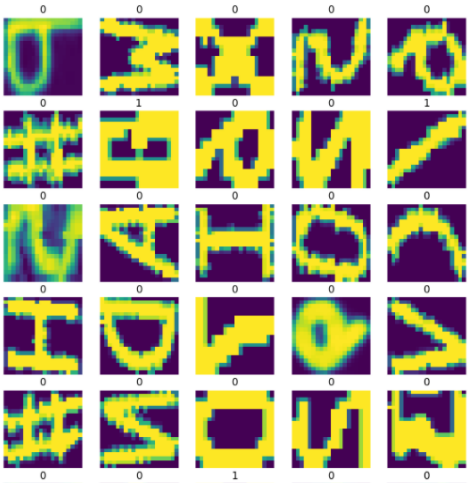

Figure 2: Image Visualisation

Data Pre-Processing and Preparation

A data sanity check was conducted to ensure cleanliness of data. No null inputs were found. 172 duplicated rows were found and removed.

The train.csv was split into 3 subsets as follows:

Subset	Training	Validation	Test
Ratio	0.72	0.08	0.20

Due to the huge imbalance in class labels of our dataset, we explore multiple oversampling techniques to balance the positive and negative classes. By oversampling, we eliminate the chances that our models will ignore the minority class during training (Pykes, 2020). The following sampling techniques will be explored and subsequent models will train with the sampling techniques.

1. Normal Sampling
2. Random Oversampling
3. ADASYN Oversampling
4. SVM-SMOTE Oversampling

Feature Engineering

Normalisation of the RGB pixel values were done to convert the integer values to floats between 0 and 1, by dividing the column values by 255. This reduces computational resources required to train the model as the multiplication of small float values allows for faster model convergence and reduces the chances of gradient explosion as compared to larger integer values (Ponraj, 2021).

Model Building and Validation

We explore multiple models to investigate which model produces the best result when tested on our held-out test set. The models investigated are summarised as follows:

Neural Networks		**Ensemble Models**	
Vanila Neural Network	Convolutional Neural Network	XGBoost	RandomForest

To reduce overfitting, the neural networks were trained with *L2 regularizer*. Model callbacks were also employed - *ReduceLROnPlateau* to allow for more efficient model convergence at the local minima, and *EarlyStopping* to prevent our model from overtraining with the test data.

Vanila Neural Network

A vanilla neural network was constructed with the following architecture (figure 3):

Input	Dense (32)	Dropout (0.3)	Dense (64)	Dropout (0.3)	Dense (128)	Dropout (0.3)	Dense (256)	Dropout (0.3)	Dense (256)	Dropout (0.3)	Output

Figure 3: Vanilla Neural Network Architecture

Results when trained with the various sampling techniques and evaluated on the held out test set are summarised as follows:

	Loss	**Accuracy**	**AUC**	**Recall**	**Precision**
Normal Sampling	0.161871	0.952689	0.933929	0.64577	0.801312
Random Oversampling	0.415842	0.812561	0.938028	0.909366	0.31835
ADASYN Oversampling	0.405759	0.818605	0.923804	0.895015	0.324036
SVM-SMOTE Oversampling	0.332876	0.865361	0.93809	0.877644	0.395507

Convolutional Neural Network

A convolutional neural network was constructed with the following architecture (figure 4):

Input, reshape	Conv (32) Batch Normalise Spatial Dropout (0.4) Max Pooling (2,2)	Conv (64) Batch Normalise Spatial Dropout (0.4) Max Pooling (2,2)	Conv (128) Batch Normalise Spatial Dropout (0.4) Max Pooling (2,2)	Conv (256) Batch Normalise Spatial Dropout (0.4) Max Pooling (2,2)	Conv (512) Batch Normalise	Flatten	Dropout (0.6) Dense (128) Dropout (0.6)	Dense (16)	Output

Figure 4: Convolutional Neural Network Architecture

Results when trained with the various sampling techniques and evaluated on the held out test set are summarised as follows:

	Loss	**Accuracy**	**AUC**	**Recall**	**Precision**
Normal Sampling	0.053670	0.983396	0.991476	0.843656	0.972150
Random Oversampling	0.062457	0.981798	0.990628	0.928248	0.880372
ADASYN Oversampling	0.072477	0.978046	0.990345	0.910876	0.858974
SVM-SMOTE Oversampling	0.069330	0.981659	0.98928	0.908610	0.893759

XGBoost

XGBoost employs an ensemble of gradient boosted decision trees to classify, with a focus on speed and performance (Brownlee, 2021). Results when trained with the various sampling techniques and evaluated on the held out test set are summarised as follows:

	Loss	**Accuracy**	**AUC**	**Recall**	**Precision**
Normal Sampling	0.075171	0.9736	0.987773	0.750755	0.952107
Random Oversampling	0.077299	0.973044	0.984978	0.84290	0.861111
ADASYN Oversampling	0.098090	0.965263	0.975970	0.817976	0.807004
SVM-SMOTE Oversampling	0.104638	0.963874	0.973277	0.813444	0.797778

RandomForest

RandomForest employs an ensemble of uncorrelated decision trees to classify. Results when trained with the various sampling techniques and evaluated on the held out test set are summarised as follows:

	Loss	**Accuracy**	**AUC**	**Recall**	**Precision**
Normal Sampling	0.111069	0.959497	0.989442	0.577039	0.970775
Random Oversampling	0.109924	0.962832	0.990123	0.623867	0.957126
ADASYN Oversampling	0.147697	0.966375	0.981978	0.756042	0.861446
SVM-SMOTE Oversampling	0.131697	0.970126	0.985507	0.760574	0.899107

Model Selection

The best performing models are evaluated based on the least binary cross-entropy loss incurred from the test data. This ensures that the model with the lowest uncertainty is selected to be evaluated on the unseen dataset.

The Convolutional Neural Network trained on normally sampled training data incurred the lowest binary cross-entropy loss among all the models explored. This model will be selected as one of the final two models for the Kaggle competition. Since the data is heavily imbalanced, the best performing model trained on oversampled training data will also be selected. The CNN model with random oversampling incurred the least binary cross-entropy loss among all the oversampled models. This model will be selected as the second model for the Kaggle competition.

	Loss	**Accuracy**	**AUC**	**Recall**	**Precision**
Normal Sampling: CNN	0.053670	0.983396	0.991476	0.843656	0.972150
Random Oversampling: CNN	0.062457	0.981798	0.990628	0.928248	0.880372

Conclusion

We have explored 4 different models with 4 different types of sampling for each model type. The Convolutional Neural Network with normal sampling performed the best. Amongst the oversampled models, CNN with random oversampling performed the best. This outcome is expected, as it was expected that the CNN model outperform the other models in all forms of sampling, as CNN models are designed to identify and isolate distinct spatial features in images to produce high confidence class predictions probabilities.

The normally sampled CNN outperformed all the oversampling methods. Across all models, random oversampling outperformed ADASYN and SMOTE-SVM. Perhaps, the synthetic generation of minority class points by the ADASYN and SMOTE-SVM algorithms were not able to accurately generate representative minority samples. The fall in precision suggests that the generated minority positive class samples are very similar to the majority negatively class samples, resulting in more false positives.

There is room for further extension of this project, such as exploring the effects of Data Augmentation to improve the performance of the model predictions. Further oversampling methods can also be explored, such as SMOTE-EN and SMOTE-Tomek, which hybridise oversampling and undersampling.

References

Brownlee, J. (2021, February 16). A Gentle Introduction to XGBoost for Applied Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Ponraj, A. (2021, February 22). A Tip A Day — Python Tip #8: Why should we Normalize image pixel values or divide by 255? | Dev Skrol. Medium. https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-8-why-should-we-normalize-image-pixel-values-or-divide-by-255-4608ac5cd26a

Pykes, K. (2020, September 10). Oversampling and Undersampling - Towards Data Science. Medium. https://towardsdatascience.com/oversampling-and-undersampling-5e2bbaf56dcf