Integration of Advanced Feature Extraction Techniques for Image-Based Insurance Cost Prediction

Akbar Bunyadzade
Istanbul Technical University
Istanbul, Turkey
bunyadzade24@itu.edu.tr

Omer Talha Demirci
Istanbul Technical University
Istanbul, Turkey
demirciom21@itu.edu.tr

Emil Huseynov
Istanbul Technical University
Istanbul, Turkey
huseynove21@itu.edu.tr

Yusa Karaslan
Istanbul Technical University
Istanbul, Turkey
karaaslanmu21@itu.edu.tr

***Abstract*—This paper discusses the development of an image classification model to predict insurance costs by integrating feature vectors extracted using state-of-the-art models such as CLIP, DINOv2, ResNet, and ViT. Our ensemble learning framework, which combines the predictions of multiple MLP classifiers through a logistic regression meta-model, achieved a Kaggle score of 0.9907 F1. The approach emphasizes techniques like learning rate decay, L2 regularization, and custom callbacks to optimize model performance. In prior attempts, simpler models and traditional ensembling methods such as stacking and basic averaging failed to achieve competitive scores, highlighting the need for advanced feature integration and weighted averaging.**

***Index Terms*—MLP, insurance cost prediction, cross-validation, stacking, ensemble learning, meta-model, weighted averaging.**

I. Introduction

The task of predicting insurance costs from images is challenging due to the high dimensionality and variability in the data. In this project, we leveraged integrated features from multiple pre-trained models *(CLIP, DINOv2, ResNet, ViT)* to train an accurate and robust classifier. Our team, *Neural Networkers*, achieved a Kaggle F1 score of 0.9907 and ranked 6th before the final announcement.

Earlier approaches, such as using single models or basic stacking methods, resulted in suboptimal performance with F1 scores stagnating around 0.987. This necessitated a shift towards more sophisticated methods, including weighted ensemble averaging and advanced regularization techniques, to maximize generalization and reduce overfitting.

II. Datasets

1) Feature Distribution Analysis: We utilized a dataset comprising 40,000 images categorized into 10 classes. Feature vectors were extracted from four pre-trained models: CLIP (512 features), DINOv2 (384 features), ResNet (2048 features), and Vision Transformer (ViT) (768 features), resulting in a combined feature set of 3712 dimensions. To understand the statistical properties of these features, we conducted a Kolmogorov-Smirnov (K-S) test to assess normality. The ViT and DINOv2 features adhered to a normal distribution ($p>0.05$), while CLIP and ResNet features deviated significantly ($p<0.05$). This analysis informed our decision to leverage all feature sets to capture diverse and complementary information, as relying solely on normally distributed features yielded only moderate baseline performances around 96 - 97% F1 scores.

2) Feature Integration and Normalization: All feature sets were horizontally concatenated to form a unified feature matrix. Given the high dimensionality of 3712 features, we employed the StandardScaler to standardize the feature vectors by removing the mean and scaling to unit variance. Although a Min-Max scaler could have been used, StandardScaler was selected because the dataset approximated a normal distribution, making it more suitable for our analysis. Meanwhile, the substantial dimensionality of the 3712 features guided us with the design of the MLP architecture, encouraging the use of a broader DNN representation. Traditional architectures, such as 512-256-128, were deemed insufficient to capture the complexity of the high-dimensional feature space in our case, necessitating the experiment of deeper and wider configurations.

III. Methods

A. Classifier Architecture

The classifier is a Multi-Layer Perceptron (MLP) with the following architecture:

- Input layer: 3712 features.
- Hidden layers:
  - Dense (1024 units, ReLU, L2 regularization $\lambda = 0.00005$).
  - Batch Normalization + Dropout ($p = 0.2$).
  - Dense (512 units) *& similar structure repeated for 256 and 128 units.*
- Output layer: Dense (10 units, Softmax activation).

1) Deep Neural Network Design: Considering the extensive feature dimensionality, we designed a Multi-Layer Perceptron (MLP) with a substantial architecture to effectively learn complex patterns from the data. The network comprises four

hidden layers with 1024, 512, 256, and 128 neurons respectively, each utilizing ReLU activation functions to introduce non-linearity. To prevent overfitting, we integrated L2 regularization ($\lambda = 5 \times 10^{-5}$), batch normalization, and dropout ($p = 0.2$) following each dense layer. The final output layer employs a softmax activation function to facilitate multi-class classification across the 10 categories.

B. Training Strategy

1) Cross-Validation and Data Augmentation: We implemented a Stratified K-Fold cross-validation approach with 5 folds and 7 distinct random seeds, resulting in 35 unique training-validation splits. Stratification ensured consistent class distribution across folds, critical for balanced performance evaluation. To enhance model generalization and mitigate overfitting, we applied the Mixup data augmentation technique with $\alpha = 0.3$. Mixup generates synthetic training samples by linearly interpolating pairs of examples and their labels, promoting smoother decision boundaries and improved robustness.

2) Optimization and Learning Rate Scheduling: The network was optimized using the Adam optimizer, chosen for its adaptive learning rate capabilities. An exponential decay learning rate schedule was employed, starting with an initial rate of 1×10^{-3} and decaying by a factor of 0.95 every 800 steps. This schedule allows the model to make significant updates initially and finer adjustments as training progresses, enhancing convergence stability.

3) Callbacks and Early Stopping: To prevent overfitting and ensure the selection of optimal model weights, we incorporated two callbacks during training:

- **Early Stopping**: Monitored validation loss with a patience of 5 epochs, halting training when no improvement was observed.
- **F1 Score Monitoring**: A custom callback tracked the Macro F1 Score on the validation set, saving model weights upon achieving a new best score.

C. Ensemble Learning and Meta-Model Stacking

1) Prediction Aggregation: Post-training, out-of-fold (OOF) predictions from all 35 models were aggregated and reshaped to form a comprehensive feature set for the meta-model. Similarly, test set predictions from each model were aggregated to construct the test feature matrix.

2) Meta-Model Configuration: We employed Logistic Regression with L2 regularization ($C = 3.0$) as the meta-model to perform ensemble stacking. This choice leverages the linear classifier's ability to effectively combine diverse predictions from multiple models, enhancing overall classification performance. The meta-model was trained on the reshaped OOF predictions, enabling it to learn optimal weightings for the base models' outputs.

D. Model Evaluation

1) Performance Metrics: The ensemble model's performance was evaluated using the Macro F1 Score, which provides an aggregate measure of performance across all classes, ensuring that each class is equally weighted regardless of its frequency.

2) Training and Testing Procedure: During training, each DNN model was fitted on the Mixup-augmented training data and validated on the corresponding validation set within each fold. The best-performing model weights based on the highest F1 Score were retained. After cross-validation, the meta-model was trained on the aggregated OOF predictions and subsequently used to generate final predictions on the test set. This two-tiered training strategy ensures robust generalization and leverages the strengths of multiple models through ensemble learning.

IV. Results and Conclusions

A. Cross-Validation Results

The model achieved the following F1 scores across the 5 folds:

- Fold 1: 0.9895
- Fold 2: 0.9902
- Fold 3: 0.9907
- Fold 4: 0.9909
- Fold 5: 0.9903

The average cross-validation F1 score was **0.9907**.

B. Kaggle Results

Earlier submissions, using basic stacking or single models, achieved scores around 0.987. The introduction of weighted averaging and improved feature integration pushed the score to **0.9907**, placing our team 6th in the leaderboard.(same points with 5th team)

References

[1] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*, 3rd ed., Packt Publishing, Birmingham, UK, 2019, Chapter on Combining Different Models for Ensemble Learning.

V. Learning Pipeline - Overview

The learning pipeline can be shown in the diagram below for a better clarification:

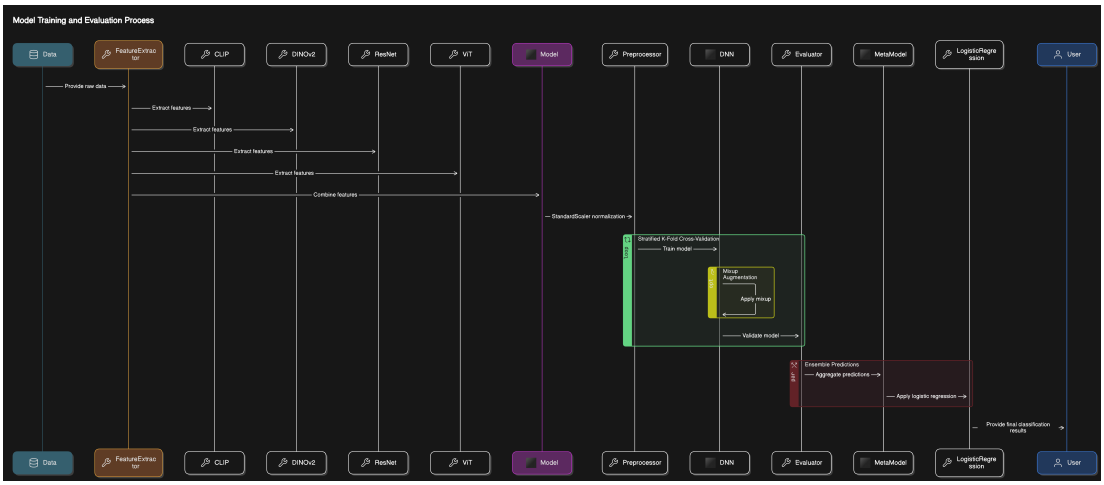

Fig. 1. Overview of the Learning Pipeline: Data preprocessing, DNN training, ensemble aggregation, and meta-model stacking.