Enhancing Classification Performance with SVM and Feature Fusion

1st Metin Ertekin Küçük
Department of Computer Engineering
Istanbul Techincal University
Istanbul, Türkiye
kucukm21@itu.edu.tr

2nd Mehmet Ali Balıkçı
Department of Computer Engineering
Istanbul Techincal University
Istanbul, Türkiye
balikci20@itu.edu.tr

3rd Ömer Faruk İşler
Department of Computer Engineering
Istanbul Techincal University
Istanbul, Türkiye
islero21@itu.edu.tr

***Abstract*—This paper presents a Support Vector Machine (SVM) based classification approach that leverages feature fusion and cross-validation to achieve high performance in a competitive Kaggle challenge. By combining CLIP and DINO features and employing a robust normalization and validation strategy, our model achieved a macro F1 score of 0.99111 and was ranked first in the competition.**

***Index Terms*—Support Vector Machine, Feature Fusion, Cross-Validation, Classification, Machine Learning**

I. Introduction

In the realm of machine learning competitions, achieving high classification performance requires meticulous feature engineering and robust model training strategies. In this project, Our objective was to develop a classifier that effectively combines multiple feature representations to enhance predictive accuracy. Utilizing Support Vector Machines (SVM) with a Radial Basis Function (RBF) kernel, we integrated CLIP and DINO features to capture both semantic and visual information from the dataset. Our approach not only improved the macro F1 score but also secured a competitive ranking in the Kaggle challenge under the team name *The AI Crew*.

II. Datasets

The project utilized two primary datasets: training and validation/testing sets. The training features were loaded from '$train\_feats.npy$', containing '$clip\_feature$' and '$dino\_feature$', while the labels were obtained from '$train\_labels.csv$'. Similarly, the validation/test features were loaded from '$valtest\_feats.npy$'.

A. Data Preprocessing

Data preprocessing involved several critical steps:

1) Feature Selection: CLIP and DINO: We employed two distinct feature extraction methods: CLIP and DINO.

CLIP (Contrastive Language–Image Pretraining) [3] is a model developed by OpenAI that learns visual concepts from natural language supervision. By leveraging a large data set of image-text pairs, CLIP can generate rich and semantically meaningful feature representations that bridge the gap between visual and textual information. This capability allows the model to capture high-level semantic features that are beneficial for classification tasks, especially in scenarios where understanding the context and relationships within the data is crucial.

DINO (Self-Distillation with No Labels) [4] is a self-supervised learning method that focuses on learning visual representations without the need for labeled data. DINO employs a teacher-student framework where the student network learns to predict the output of the teacher network under different augmentations. This approach enables the extraction of robust and invariant features that are effective in capturing fine-grained visual details and patterns within the dataset.

2) Rationale for Using Both CLIP and DINO Features: The combination of CLIP and DINO features was strategically chosen to leverage the strengths of both models:

- **Semantic Richness**: CLIP provides high-level semantic features derived from natural language supervision, enabling the model to understand and interpret complex relationships and contexts within the data.
- **Visual Robustness**: DINO offers robust visual features learned through self-supervision, capturing intricate visual patterns and details that might be overlooked by purely supervised methods.
- **Complementary Information**: By fusing CLIP and DINO features, the model benefits from a comprehensive feature set that encompasses both semantic and visual information, enhancing the overall discriminative power and improving classification performance.

3) Feature Fusion and Normalization: The CLIP and DINO features were horizontally stacked to form a unified feature matrix, ensuring that each feature type contributes equally to the model. To maintain consistency and prevent any single feature from dominating due to scale differences, we applied standard scaling using *StandardScaler*, fitting it on the training data and transforming both the training and validation/test datasets accordingly.

B. Handling Missing Data

The "$loaddata$" function incorporated error handling to manage possible missing files, ensuring robustness in data loading. If a file was not found, the function would notify the user and allow the workflow to continue without interruption.

III. METHODS

Our classification pipeline consists of the following components:

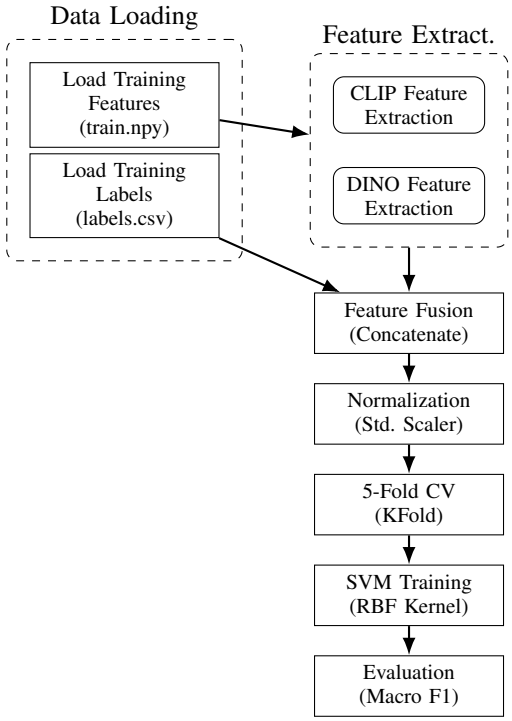

Fig. 1: Learning Pipeline

A. Feature Engineering

We combined CLIP and DINO features to leverage both textual and visual representations. This fusion was achieved through horizontal stacking, resulting in a unified feature vector that encapsulates multifaceted data characteristics.

B. Model Selection

We selected the Support Vector Machine (SVC) with an RBF kernel for several compelling reasons:

1) Effectiveness in High-Dimensional Spaces: SVMs are particularly well suited for high-dimensional data, which is often the case when combining multiple feature sets such as CLIP and DINO. The ability of SVMs to handle large feature spaces without overfitting is crucial for maintaining model performance and generalization.

2) Handling Non-Linear Relationships: The Radial Basis Function (RBF) kernel enables the SVM to capture non-linear relationships within the data. Given the complexity of the features derived from CLIP and DINO, which encapsulate both semantic and visual information, the RBF kernel allows the model to create more flexible decision boundaries compared to linear models.

3) Robustness to Overfitting: With appropriate hyperparameter tuning, SVMs can achieve a balance between bias and variance, reducing the risk of overfitting. The regularization parameter C controls the trade-off between achieving a low training error and a low testing error, while the γ parameter influences the decision boundary's smoothness.

4) Scalability and Efficiency: Despite their effectiveness, traditional SVMs can be computationally intensive for very large datasets. However, given the dataset size in this project, SVMs remained a viable and efficient choice. Additionally, leveraging the *scikit-learn* implementation of SVMs provides optimized performance and scalability.

5) Interpretability and Theoretical Foundation: SVMs have a strong theoretical foundation in statistical learning theory, providing guarantees on generalization performance. The concept of support vectors offers insights into which data points are critical for defining the decision boundary, aiding in model interpretability.

6) Comparative Performance: During preliminary experiments, SVMs demonstrated superior performance compared to other classifiers such as Logistic Regression and Random Forests, particularly in terms of macro F1 scores. This empirical evidence further reinforced the decision to employ SVMs for the final model.

7) Hyperparameter Tuning: The chosen hyperparameters were meticulously selected to optimize model performance:

- **C**: Set to 500 to impose a higher penalty for misclassification, encouraging a more precise decision boundary.
- **Gamma**: Set to "scale" to allow the kernel coefficient to adjust based on the number of features, promoting adaptability.
- **Random State**: Fixed at 1 to ensure reproducibility of results.

In summary, the selection of SVM with an RBF kernel was driven by its ability to effectively handle high-dimensional, non-linear data, its robustness against overfitting, and its strong theoretical underpinnings. These attributes made SVM an optimal choice for leveraging the fused CLIP and DINO features to achieve high classification performance in this project.

C. Training and Validation

We employed 5-fold cross-validation using *KFold* with shuffling and a fixed random state to ensure consistent splits. In each fold:

1) The model was trained on the training split.
2) Predictions were made on the validation split.
3) The macro F1 score was computed to evaluate performance.

The mean F1 score across all folds provided an estimate of the model's generalization capability.

IV. Results and Conclusions

A. Cross-Validation Performance

Our 5-fold cross-validation yielded the following macro F1 scores:

- Fold 1 Score: 0.9893
- Fold 2 Score: 0.9906
- Fold 3 Score: 0.9892
- Fold 4 Score: 0.9891
- Fold 5 Score: 0.9899

The mean F1 score across all folds was 0.9896, demonstrating consistent performance and robustness of our model.

B. Competition Outcome

Upon final evaluation on the validation/test set, our model achieved a Kaggle score of 0.99111, securing the 1st rank out of 60 participating teams under the Kaggle as *The AI Crew*.

In conclusion, the integration of multiple feature sets and the application of SVM with hyperparameter tuning significantly enhanced our classification performance, validating the effectiveness of our approach in competitive settings.

References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995.

[2] J. Park, K. Kim, and S. Lee, "Feature Fusion Techniques for Image Classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 3, pp. 567-580, March 2020.

[3] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," *arXiv preprint arXiv:2103.00020*, 2021.

[4] M. Caron et al., "Emerging Properties in Self-Supervised Vision Transformers," *arXiv preprint arXiv:2104.14294*, 2021.

[5] S. Scott, "Support Vector Machines for Classification and Regression," *Journal of Machine Learning Research*, vol. 5, pp. 45-66, 2004.