Hyperparameter Tuning and Feature Transformation Analysis in Machine Learning Models

Introduction

In this project, we aimed to train three types of classifiers on the CIFAR-10 dataset to explore how different hyperparameters and feature transformations influenced performance. The dataset consists of $60,000$ images with 10 mutually disjoint classes, mostly consisting of animals and vehicles. The training set contains $50,000$ images and the testing set $10,000$, and both sets contain balanced classes. We selected logistic regression, support vector machines (SVM), and convolutional neural networks (CNN) as our classifiers. Our analysis focused on balancing overfitting and underfitting while optimizing key metrics like F1 score, accuracy, and ROC-AUC.

Unsupervised Analysis

Although the images only have a 32×32 resolution and 3 channels, the resulting feature count of 3072 is too high for direct analysis of feature correlation and relationship with the class label. Therefore, we reduced the data's dimensionality through principal component analysis (PCA). Firstly, we analyze how much variance in the original data is retained after applying PCA with 1 to 100 components.

Table 1: Variance Captured by PCA Components

# Components	1	2	3	4	5	10	20	50	100
Variance Kept	0.291	0.403	0.470	0.507	0.543	0.655	0.745	0.843	0.901

From the leveling-off trend of PCA maintained variance, we decided to use 100 components for our unsupervised analysis and as a feature transformation for supervised learning later on. This preserves more than 90% of the original data's variance. Also of note is that the first PCA component is responsible for the majority of variance captured, regardless of the total number of components.

Furthermore, we observe that the PCA-transformed features do not have correlations amongst themselves or with the target. The distribution of PCA-transformed features are not skewed.

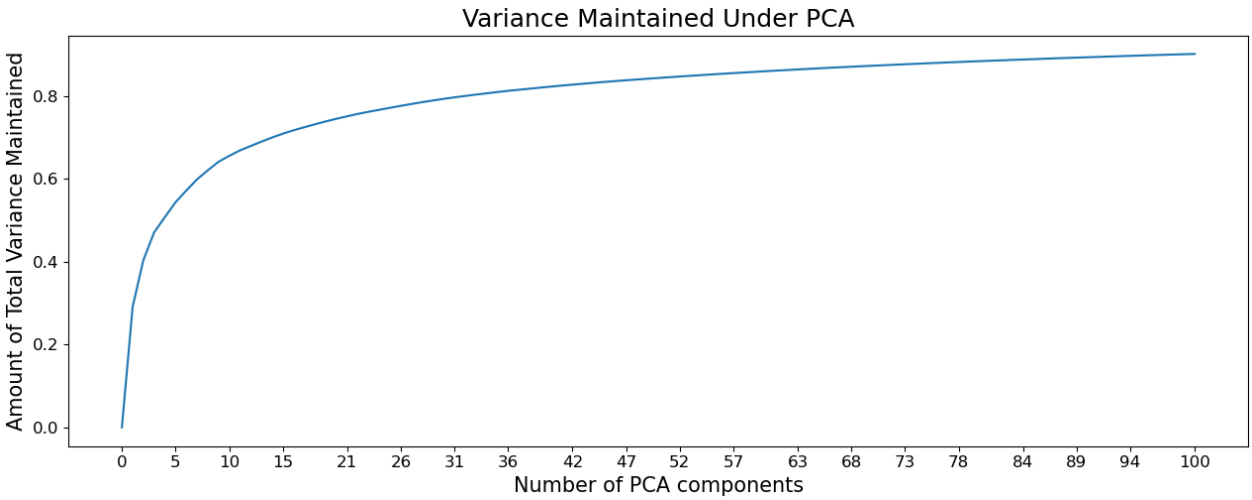
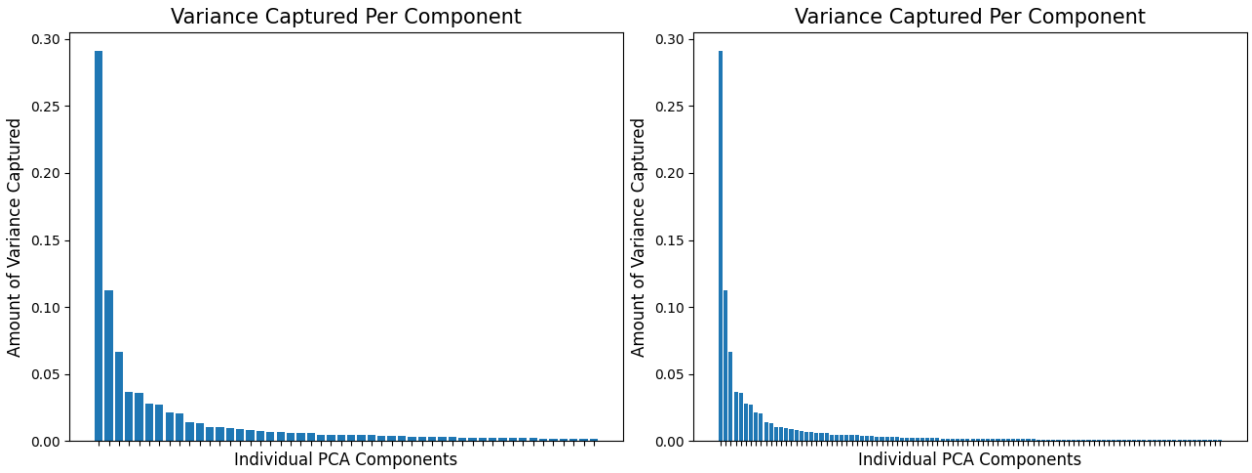


Figure 1: Amount of original data's variance maintained by PCA with different number of components



(a) 50 PCA Components

(b) 100 PCA Components

Figure 2: The contribution of individual PCA components towards maintaining original data's variance

Supervised Learning

Logistic Regression

We fitted a logistic regression model on our dataset with scikit-learn's multinomial strategy for our multi-class task. In additional to training on original data, we applied three feature transformations and performed grid search on both the regularization type (penalty) and its weighing coefficient C. The experimental configurations are summarized in the table below. Note that every combination of the

The regularization parameter (param_C) had a noticeable impact on Logistic Regression's

Table 2: Configuration for Training Logistic Regression

Hyperparameters	Values in Grid Search						
C	0.0001	0.0005	0.001	0.005	0.01	1	(None)
penalty	L1			L2			(None)
feature transformations	PCA (100 components)		PCA & RBF (gamma=0.1)		PCA & Polynomial (degree 2)		(None)

Table 3: Add caption

feature transformations	PCA& Polynomial	None	PCA	PCA& RBF	PCA& RBF	PCA& Polynomial	PCA	None
C	0.0001	0.0005	1	0.01	0.0001	0.0001	0.0001	0.0001
penalty	l2	l2	l1	l1	l1	l1	l1	l1
accuracy	0.5113	0.4001	0.3930	0.1010	0.1000	0.1000	0.1000	0.1000
acccuracy_train	0.7300	0.4597	0.4064	0.1139	0.1000	0.1000	0.1000	0.1000
precision_micro	0.5113	0.4001	0.3930	0.1010	0.1000	0.1000	0.1000	0.1000
recall_micro	0.5113	0.4001	0.3930	0.1010	0.1000	0.1000	0.1000	0.1000
f1_micro	0.5113	0.4001	0.3930	0.1010	0.1000	0.1000	0.1000	0.1000
f1_macro	0.5087	0.3964	0.3891	0.0888	0.0182	0.0182	0.0182	0.0182
jaccard_micro	0.3434	0.2501	0.2446	0.0532	0.0526	0.0526	0.0526	0.0526
jaccard_macro	0.3463	0.2503	0.2446	0.0469	0.0100	0.0100	0.0100	0.0100
precision_macro	0.5114	0.3956	0.3882	0.1052	0.0100	0.0100	0.0100	0.0100
recall_macro	0.5113	0.4001	0.3930	0.1010	0.1000	0.1000	0.1000	0.1000
roc_auc_ovr	0.8704	0.8142	0.8113	0.4979	0.5000	0.5000	0.5000	0.5000
roc_auc_ovo	0.8704	0.8142	0.8113	0.4979	0.5000	0.5000	0.5000	0.5000

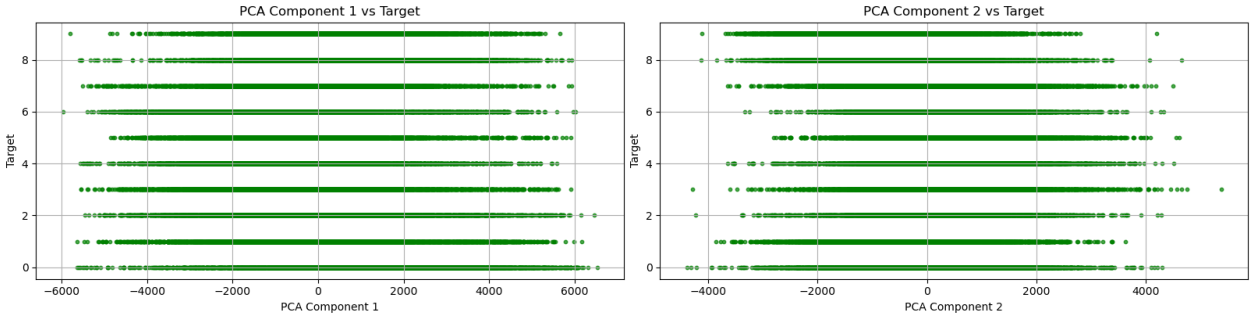


Figure 3: Relationships between the target and the first and second PCA features

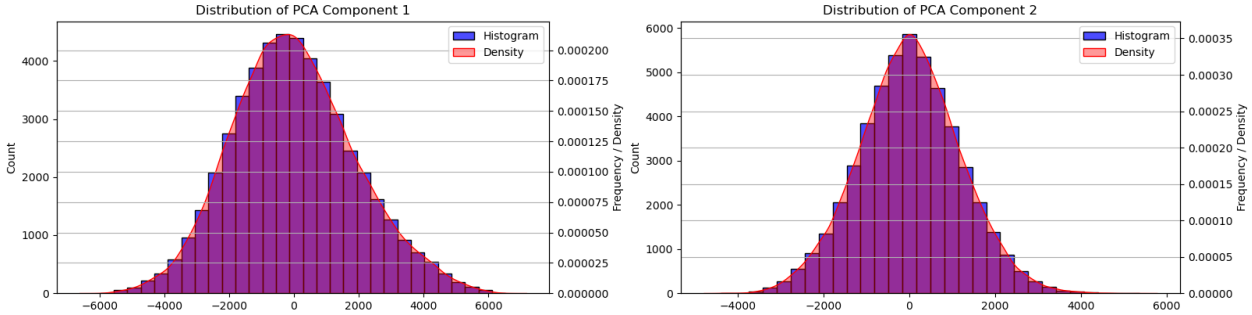


Figure 4: Distribution of the first and second PCA features

performance:

- When param_C was too small (indicating strong regularization), the model was overly constrained, resulting in underfitting and low performance across metrics like accuracy and F1 score.
- Increasing param_C to a moderate range allowed the model to balance bias and variance effectively, leading to its best results.
- When param_C became too large (weak regularization), the model overfit the training data, performing well on it but failing to generalize to unseen data.

Regularization type also played an important role:

- **L1 Regularization**: Ideal for sparse datasets, as it eliminates less important features, simplifying the model.
- **L2 Regularization**: Better suited for datasets where all features contribute meaningfully, as it penalizes large coefficients uniformly. Based on the table above,we can easily determied that the L2 regularization has a better performance for out dataset.

Table 4: Configuration for Training SVM

Hyperparameters	Values in Grid Search					
C	0.0001	0.0005	0.001	0.005	0.01	1
gamma	scale			auto		
feature transformations	(None)		PCA (100 components)		PCA & Polynomial (degree 2)	

Table 5: Add caption

feature transformation	PCA	(None)	PCA& Polynomial	(None)	PCA& Polynomial	PCA
C	1	1	1	0.005	0.005	0.0001
Gamma	scale	auto	auto	scale	scale	scale
accuracy	0.5398	0.4941	0.4568	0.3113	0.1703	0.2801
accuracy train	0.8068	0.7158	0.8825	0.3183	0.2044	0.2913
precision_micro	0.5398	0.4941	0.4568	0.3113	0.1703	0.2801
recall_micro	0.5398	0.4941	0.4568	0.3113	0.1703	0.2801
f1_micro	0.5398	0.4941	0.4568	0.3113	0.1703	0.2801
f1_macro	0.5390	0.4925	0.4561	0.2982	0.0871	0.2546
jaccard_micro	0.3697	0.3281	0.2960	0.1843	0.0931	0.1628
jaccard_macro	0.3748	0.3312	0.3009	0.1780	0.0485	0.1505
precision_macro	0.5406	0.4935	0.4889	0.3131	0.3593	0.3617
recall_macro	0.5398	0.4941	0.4568	0.3113	0.1703	0.2801

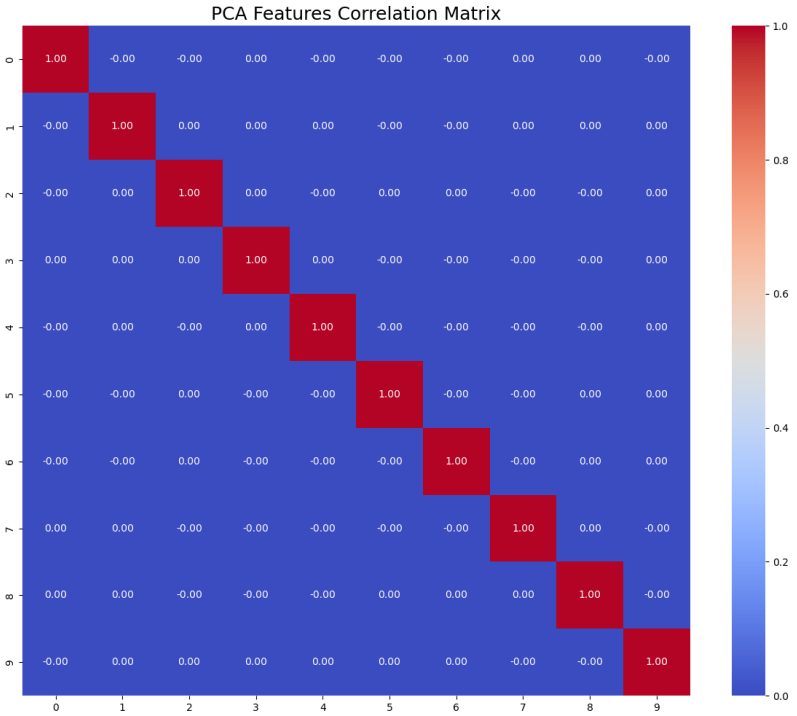


Figure 5: PCA-transformed features are independent from each other

Support Vector Machines (SVM)

SVM performance was heavily influenced by the kernel coefficient (param_gamma) and regularization parameter (param_C):

- When param_gamma was too low, the model's decision boundary was overly simplistic, leading to underfitting and poor predictions.
- A high param_gamma resulted in overly complex decision boundaries that overfit the training data.
- Moderate values of param_gamma struck the right balance, allowing the model to capture complex relationships without overfitting.

Similarly, the regularization parameter (param_C) controlled the trade-off between simplicity and complexity:

- Small param_C values created simple models that underfit the data.
- Large param_C values made the model too complex, increasing variance and overfitting.

Using the RBF kernel further improved SVM performance by capturing non-linear relationships in the data, significantly enhancing metrics like F1 and ROC-AUC. Based on the above table, we can see change in the regularization parameter can determine the whole performance even for the same kind of feature transformation.(i.e.with PCA can reach a test accuracy of 0.5398 with parameter 1, but if the parameter is 0.0001, the accuracy is only 0.2801), A general pattern of SVM seems to be higher regularization parameter.

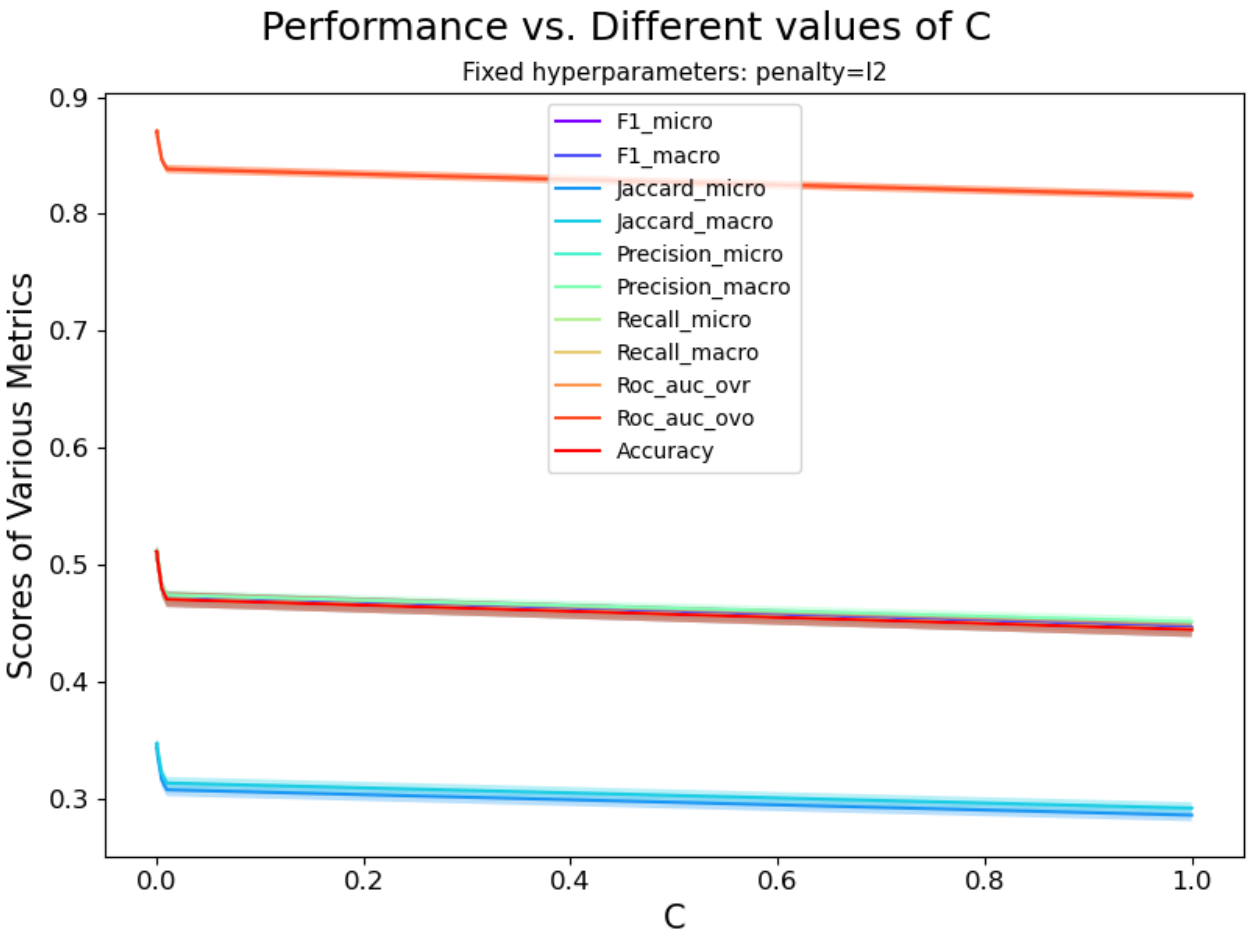


Figure 6: Influence of hyperparameter C on logistic regression performance (with PCA & Polynomial transformations)

Convolutional Neural Networks (CNN)

For CNNs, hyperparameters like learning rate (param_lr), dropout rate (param_module__drop_rate), and weight decay (param_optimizer__weight_decay) had a strong influence on model performance:

- **Learning Rate**: A low learning rate caused the model to converge too slowly, while a high learning rate made training unstable. A moderate learning rate (0.001–0.01) allowed the model to converge efficiently without instability.
- **Dropout Rate**: Low dropout rates (< 0.2) led to overfitting, as the model memorized the training data. High dropout rates (> 0.5) caused underfitting by reducing the model's capacity to learn. Dropout rates between 0.2 and 0.5 provided the best generalization.
- **Weight Decay**: Weight decay added regularization by penalizing large weights. Moderate values (0.0001–0.01) improved generalization without overly constraining the

Table 6: Configurations for Training CNN

Hyperparameters	Values in Grid Search			
Learning Rate	0.005	0.01		
Dropout	0	0.1	0.2	0.3
Momentum	0	0.45	0.9	
L2 Regularization	0	0.001	0.005	0.01
Model Type	Resnet18	Restnet34	Resnet50	

Table 7: Add caption

Model Types	resnet34	resnet18	resnet50	resnet34	resnet50	resnet18
learning_rates	0.005	0.005	0.005	0.005	0.005	0.005
dropout	0.3	0.3	0.2	0	0	0
momentum	0.9	0.9	0.9	0	0	0
l2_reg	0.005	0.005	0.005	0.001	0	0
accuracy	0.7454	0.7333	0.7250	0.4852	0.4863	0.5403
accuracy_train	0.9465	0.9316	0.8851	0.5799	0.5813	0.6444
precision_micro	0.7454	0.7333	0.7250	0.4852	0.4863	0.5403
recall_micro	0.7454	0.7333	0.7250	0.4852	0.4863	0.5403
f1_micro	0.7454	0.7333	0.7250	0.4852	0.4863	0.5403
f1_macro	0.7459	0.7328	0.7230	0.4841	0.4865	0.5394
jaccard_micro	0.5941	0.5789	0.5694	0.3203	0.3213	0.3702
jaccard_macro	0.6023	0.5846	0.5741	0.3228	0.3257	0.3732
precision_macro	0.7513	0.7389	0.7311	0.4844	0.4878	0.5392
recall_macro	0.7454	0.7333	0.7250	0.4852	0.4863	0.5403
roc_auc_ovr	0.9560	0.9535	0.9523	0.8633	0.8613	0.8937
roc_auc_ovo	0.9560	0.9535	0.9523	0.8633	0.8613	0.8937

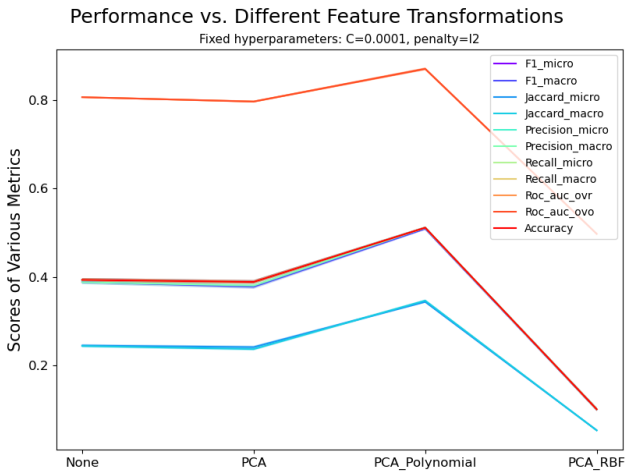


Figure 7: Influence of feature transformations on logistic regression performance (with PCA & Polynomial transformations)

model. High weight decay led to underfitting, limiting the model's ability to capture complex patterns.

For CNN, based on the above table, we can see the ones with higher dropout and momentum rate always have higher accuracy, also adding regularization is also a crucial step for better performance.

Overfitting vs. Underfitting

We observed overfitting in models with insufficient regularization or dropout, where performance on training data was excellent but failed to generalize to test data. This was particularly evident in CNNs with low dropout rates and SVMs with high param_gamma. On the other hand, underfitting occurred when models were overly constrained, such as with high dropout rates, low param_gamma, or excessive weight decay. The optimal performance came from carefully tuning hyperparameters to balance these extremes.

Bias-Variance Trade-Off

The bias-variance trade-off was clear across all models:

- **High Bias**: Strong regularization, low param_gamma, or excessive weight decay limited the model's ability to learn patterns, resulting in underfitting.
- **High Variance**: Weak regularization, low dropout, or high param_gamma made the model overly sensitive to training data, causing overfitting.

The best results were achieved when hyperparameters minimized both bias and variance, ensuring the models generalized effectively to unseen data.

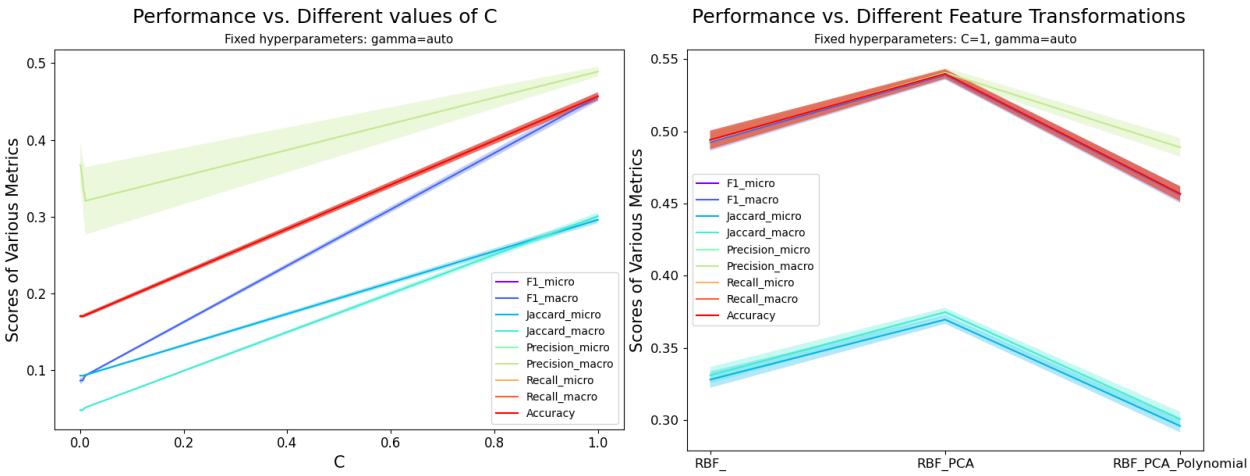


Figure 8: Influence of experimental configurations on SVM performance (with PCA & Polynomial transformations)

Figure 9: Influence of feature transformations on SVM performance (with PCA & Polynomial transformations). Here "RBF" references SVM stil internally using the RBF kernel.

Key Takeaways

- **Logistic Regression**:
  - Moderate param_C values yielded the best balance between bias and variance.
  - L1 regularization was ideal for sparse datasets, while L2 performed better on balanced datasets.
- **Support Vector Machines (SVM)**:
  - Moderate param_gamma and param_C values were critical for optimal performance.
  - Using the RBF kernel significantly improved the model's ability to capture nonlinear relationships.
- **Convolutional Neural Networks (CNN)**:
  - Learning rates between 0.001 and 0.01, dropout rates of 0.2 to 0.5, and moderate weight decay (0.0001–0.01) ensured the best generalization.

1 Model Performance Analysis

This section presents a comparative analysis of three models: Logistic Regression, ResNet, and SVM with RBF Kernel. The evaluation metrics include Best Test Accuracy, Best F1 Score, and the Lowest Standard Deviation of Test Accuracy.

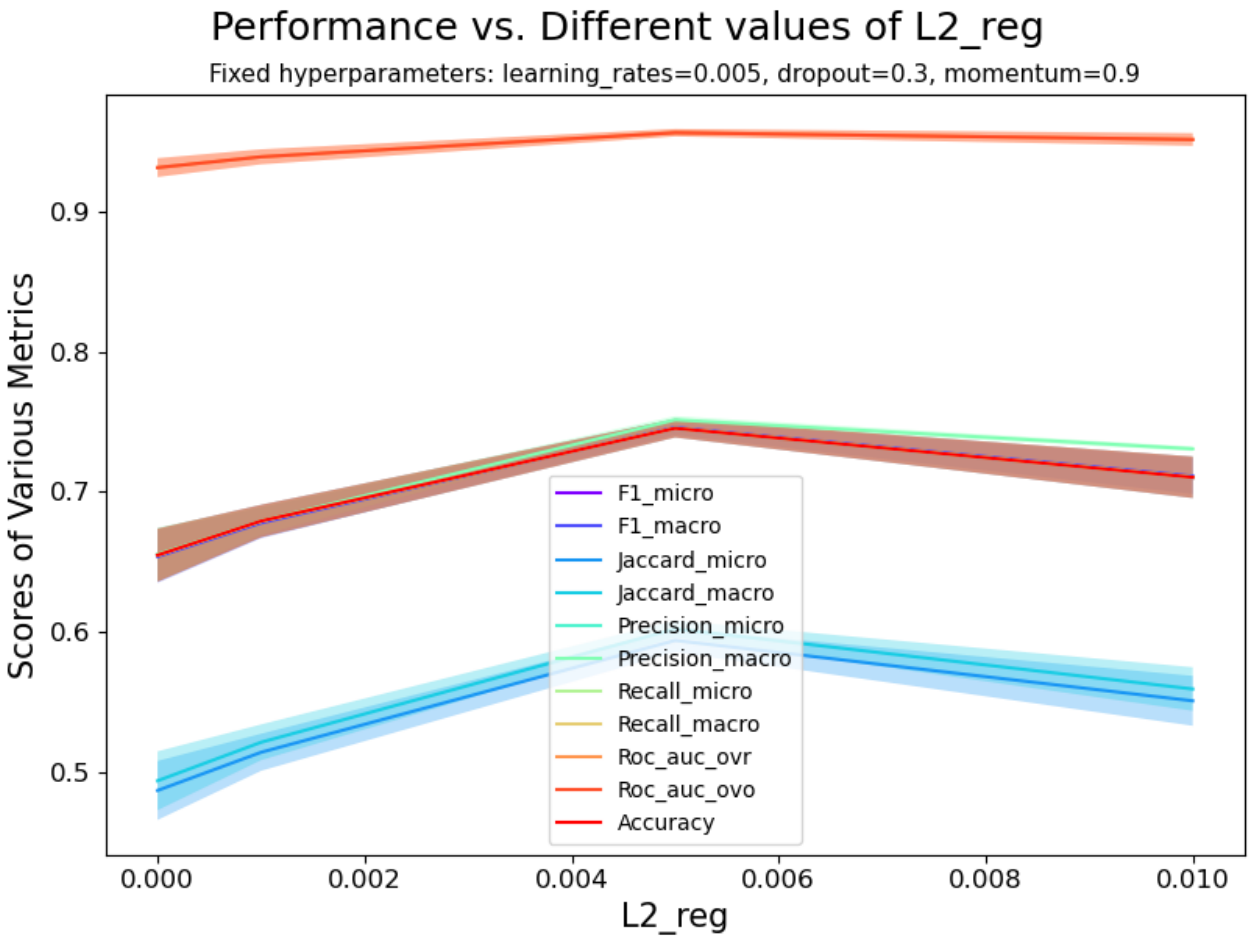


Figure 10: Influence of the amount of L2 regularization on the performance of Resnet34

Model	**Best Test Accuracy**	**Best F1 Score**	**Lowest Std Dev (Test Accuracy)**
Logistic Regression	0.51125	0.51125	0.00000
ResNet	0.74536	0.74536	0.00119
SVM (RBF Kernel)	0.53983	0.53983	0.00165

Table 8: Comparison of Model Performance Metrics.

The ResNet model outperformed the Logistic Regression and SVM models, achieving the highest Test Accuracy (0.74536) and F1 Score (0.74536). Additionally, it demonstrated greater stability with the lowest standard deviation in Test Accuracy (0.00119).

Conclusions

Conclusions and Analysis

- Based on the chart:
  - ResNet architectures (ResNet18, ResNet34, ResNet50) significantly outperform

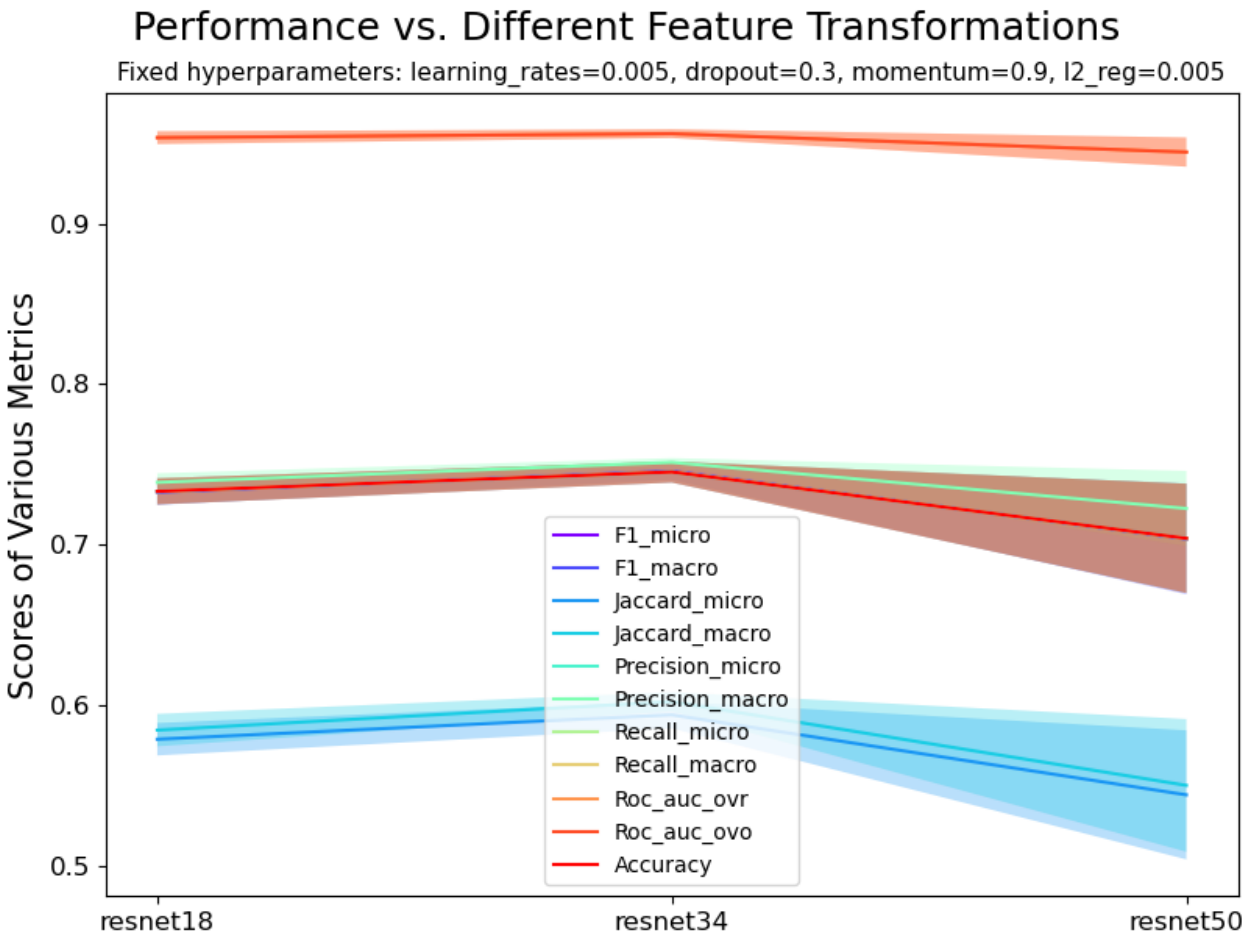


Figure 11: Influence of using Resnet34's best parameters on all three Resnet architectures tested

logistic regression and SVM models across most metrics.

- The highest accuracy and ROC-AUC scores are observed with ResNet34, indicating its superior ability to generalize and distinguish between classes.
- Logistic regression with PCA or polynomial features shows better performance than RBF-based transformations, but still lags behind more advanced models like ResNet.
- SVM models show moderate performance, with SVM_RBF achieving the best results among the SVM configurations.

- Insights from the results:
  - ResNet models are particularly effective for complex image data due to their deep feature extraction capabilities.
  - Logistic regression and SVMs, being less complex, are more suited to smaller, less complex datasets.

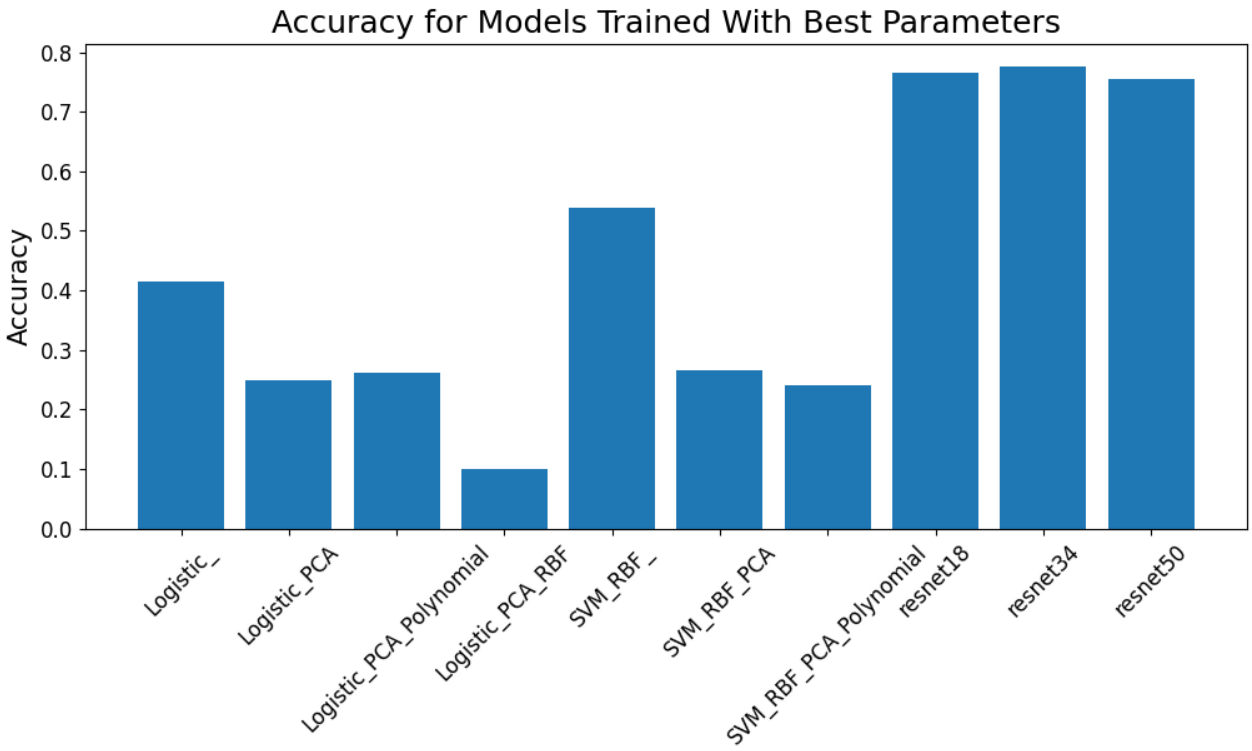


Figure 12: Accuracy on test set from models trained with the best parameters

  - Feature transformations (PCA, polynomial features) improve performance for simple models but cannot compete with deep learning approaches.

Recommendations

- Use L1 regularization for feature selection in sparse datasets and L2 for balanced datasets.
- Carefully tune param_gamma and use RBF kernels for SVMs when working with non-linear data.
- For CNNs, combine moderate dropout rates, learning rates, and weight decay with a dynamic learning rate schedule to achieve robust performance.

This analysis highlights the importance of systematic hyperparameter tuning in achieving optimal model performance. By balancing complexity and generalization, these methods can be applied to other machine learning tasks to ensure reliable and accurate predictions.

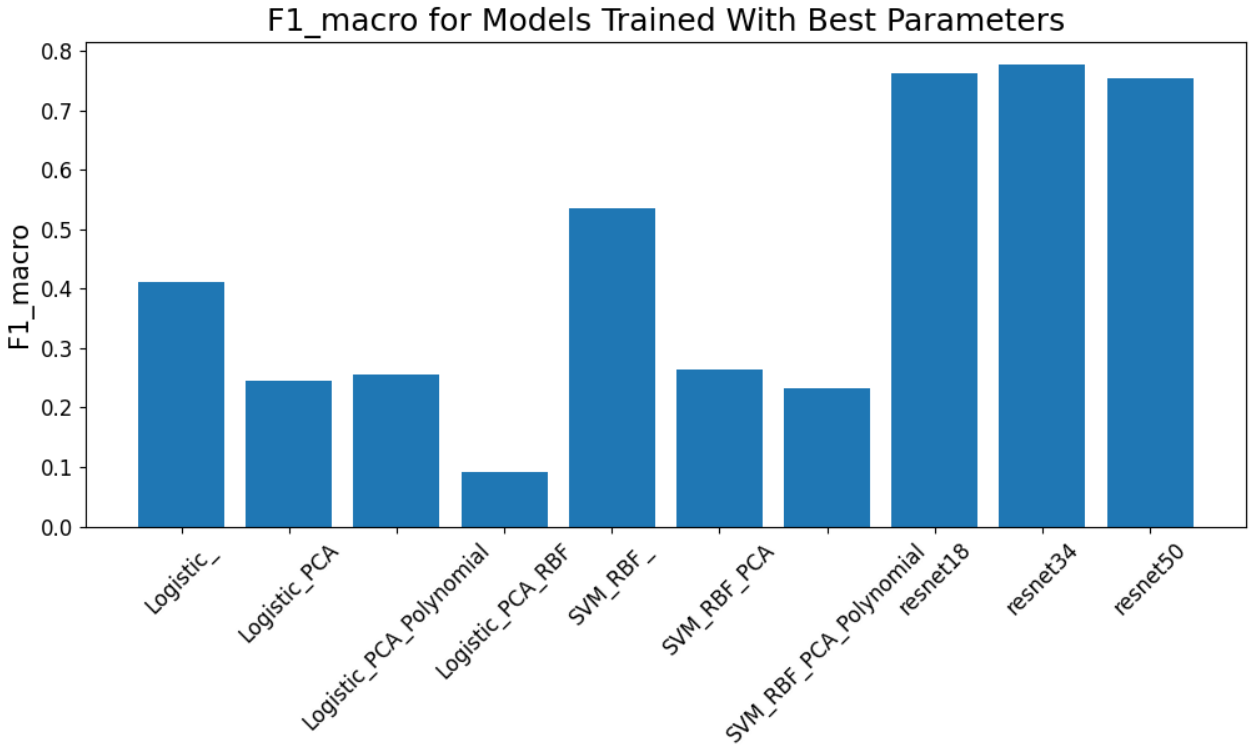


Figure 13: F1 (macro) score on test set from models trained with the best parameters. The F1 micro score is virtually identical due to balanced classes.

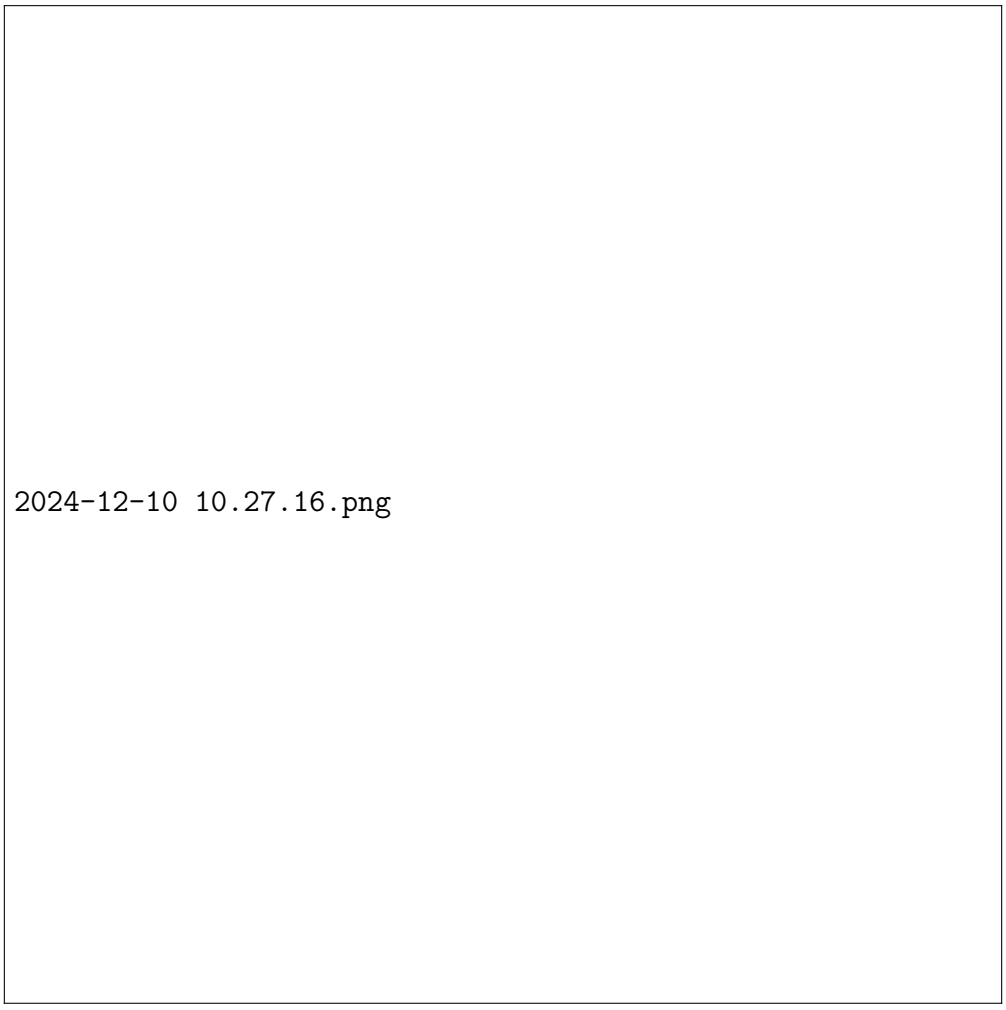


Figure 14: Comparison of Models Across Various Metrics