Deep Learning - Final Project

Enhancing Image Classification with Deep Learning: A Comparative Study of CNN Architectures, Data Augmentation and Hyperparameter Tuning

By Shelly Levy 318901550 and Adam Bublil 207271610

https://github.com/shellylevy/Enhancing_Image_Classification_with_Deep_Learning

Keywords
Deep Learning, Image Classification, CNN, Data Augmentation, Hyperparameter Tuning, Neural Networks, Machine Learning, PCA, Max Pooling, Classification Performance, Dataset Balancing.

ABSTRACT -
This project explores methods for enhancing image classification performance using deep learning models, focusing on Convolutional Neural Networks (CNNs), data augmentation, and hyperparameter tuning. We experimented with different CNN architectures, comparing a baseline model with an improved CNN version featuring additional convolutional layers and dropout regularization. Data augmentation techniques, including horizontal flips, rotations, and color jittering, were applied to expand the dataset, while undersampling was used to assess model performance on reduced data. Hyperparameter tuning optimized learning rates, dropout rates, and weight decay values to improve generalization.

1. INTRODUCTION

Image classification is a core task in computer vision with applications in fields such as autonomous driving, security, and healthcare. Convolutional Neural Networks (CNNs) have demonstrated exceptional performance in this domain by automatically extracting spatial features from images. However, achieving high accuracy requires optimizing multiple aspects of the learning pipeline, including model architecture, dataset modifications, and hyperparameter selection.
This study investigates the impact of architectural improvements, data augmentation, and hyperparameter tuning on CNN performance in a binary classification task distinguishing between "Bike" and "Car" images. The dataset, sourced from Kaggle, consists of 3,996 labeled images. To improve classification accuracy, we tested various techniques such as data augmentation, undersampling, and hyperparameter tuning. We also introduced an enhanced CNN2 model with additional convolutional layers and dropout regularization. By comparing these approaches, we aim to determine the best combination of techniques for optimizing CNN-based image classification.

2. LITERATURE REVIEW

To establish a methodological framework and gain insights for our research, we have sought related articles. Although we have not found a completely aligned article for our project's focus, the study "MULTI PRETEXT TASK SELF-SUPERVISED LEARNING FOR CULTURAL HERITAGE CLASSIFICATION" does offer relevant insights into the application of self-supervised learning and transfer learning for image classification. Its examination of SSL via pretext tasks such as image rotation and jigsaw puzzle permutations is pertinent to our investigation into the effects of image permutation on classification performance. Specifically, it aids in comprehending how neural networks, trained on permuted images, might be optimized to achieve improved classification accuracy.
We've identified several key insights from the research that we aim to apply to our own study:
Dataset and Data Augmentation: The research utilized augmentation techniques such as rotation, jigsaw puzzle permutation, and image recreation from inpainted images to expand the training dataset. This approach enhanced the model's performance. We plan to employ similar augmentation strategies to increase our dataset, potentially improving our models's

robustness and accuracy.
CNN Architecture: Various CNN architectures were explored in the research, including a baseline model, a deep bottleneck model inspired by ResNet50, and a transfer learning model using ResNet50. The superior performance of the transfer learning model indicates that we should consider this approach. Also found that deeper and more complex network architectures might yield better results - thus, in our investigation, it would be prudent to experiment with varying depths and complexities in our CNN models to determine the optimal configuration for our specific task.
Performance Evaluation: The study's evaluation metrics included accuracy, top-1 error, and top-5 error rates; these two not be directly applicable to our binary classification scenario, precision-recall analysis and confusion matrices. These metrics will be essential in assessing the impact of image permutation on our classification models.

3. METHODOLOGY

In this project, the dataset used contains a collection of 3996 images, with each image classified into two categories: "Bike" and "Car." The images are sourced from a public repository, specifically a Kaggle dataset, which provides labeled examples for classification tasks.

3.1 EDA

The images come in various formats, including JPEG, PNG, and WEBP, with the majority being in JPEG format (3969 images), followed by a few in PNG (25 images) and WEBP (2 images). The images exhibit a variety of sizes and resolutions, with 429 unique image sizes observed across the dataset.
Given that the task is to classify images into two categories, we focus on both the content (bike vs. car) and various characteristics of the images, such as their size, pixel intensity, brightness, and sharpness. These characteristics are key to understanding the dataset's properties and improving the performance of the classification model.

Image Distribution by Category

As part of the initial EDA, we observe the distribution of the dataset's labels (Bike vs. Car). Out of the total 3996 images, 1999 images belong to the "Bike" class, and 1997 images belong to the "Car" class. This balance is relatively even, which is ideal for training a machine learning model that can generalize well to both classes.

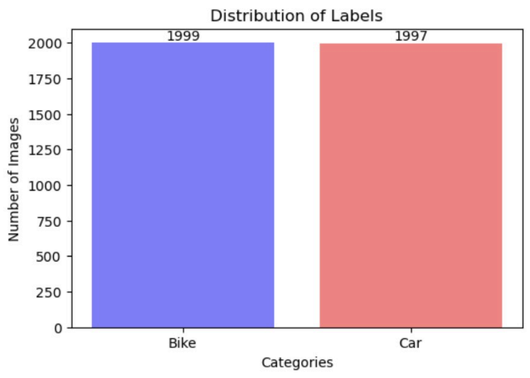
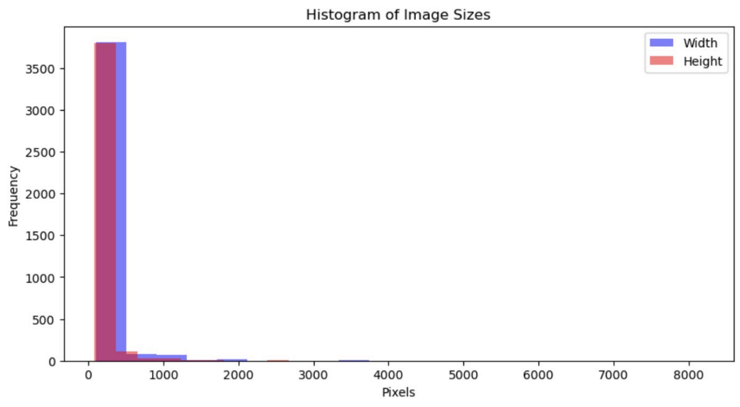


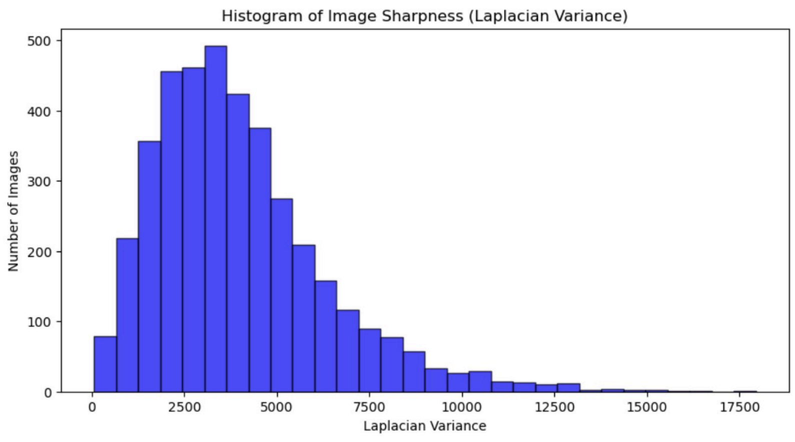
Image Size Distribution

The images in the dataset have varying sizes, with 429 unique dimensions. The histogram of width and height shows that most images have relatively small dimensions, with both width and height values clustering between 0 and 1000 pixels. There are very few images with dimensions exceeding 4000 pixels, indicating that the images are mostly smaller and manageable for analysis and model training.



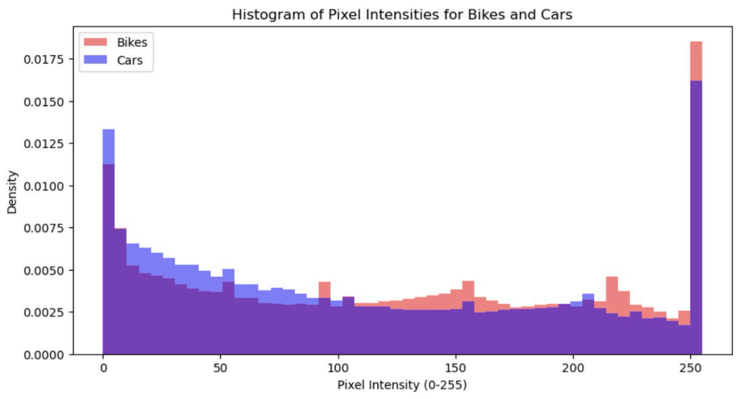
Sharpness Analysis

We also evaluated the sharpness of the images using the Laplacian variance method, which measures the amount of variation in pixel intensity to assess image sharpness. The histogram of sharpness shows a fairly uniform distribution, with most images falling between 2000 and 5000 in Laplacian variance. This suggests that most images in the dataset are neither excessively blurry nor perfectly sharp, but exhibit a moderate level of sharpness suitable for classification tasks.



Pixel Intensity Distribution

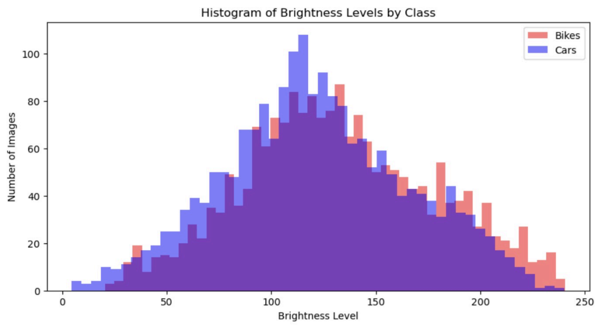
A key part of image analysis involves understanding the pixel intensities, which are visualized in a histogram for both bike and car images. The histogram of pixel intensities reveals that the pixel intensity for car images tends to have a higher concentration of values near 250 (indicating brighter images), whereas bike images have a more varied distribution with a peak in the lower intensity range. This difference could potentially help the model differentiate between the two classes based on pixel intensity patterns.

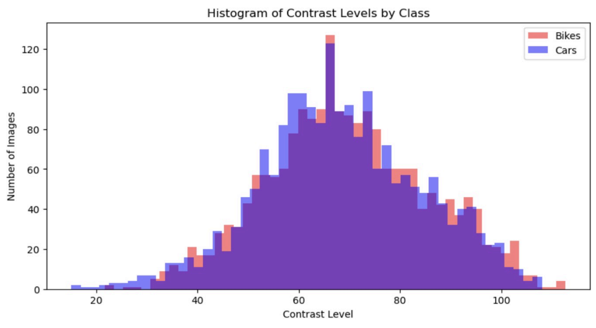


Brightness and Contrast Distribution

In addition to pixel intensity, brightness and contrast are important features for image classification. Histograms of brightness levels for both classes reveal that bike images tend to have higher brightness levels compared to car images. A separate histogram of contrast levels shows that bike images exhibit greater contrast variability, whereas car images tend to have more consistent contrast levels.

These observations highlight the need to account for such features in preprocessing and feature engineering, as they might significantly impact model performance.





To better understand the characteristics of low, medium, and high brightness levels, we included sample images from both classes, displaying a "low brightness" car, a "medium brightness" car, and a "high brightness" bike. This visual aid underscores the variance in brightness across the dataset and its potential importance in classification.

3.2 Data Preprocessing -

In the Data Preprocessing phase, several steps were taken to ensure the dataset was cleaned, balanced, and appropriately prepared for training:

3.2.1 Data Cleaning:

Identifying Broken Images:
To maintain data integrity, we aimed to identify broken images with missing data or corrupted pixels. This ensures that only high-quality images are retained, preventing errors in further processing or analysis. No broken images were found and removed from the dataset.
Identifying and Removing Duplicates:
211 duplicate images were detected and removed using the dHash method. This method identifies images with identical or highly similar content, ensuring the dataset remains diverse and preventing the model from learning redundant patterns.
Identifying Anomalous Images:
We analyzed the aspect ratios of the images and used an outlier detection approach (Isolation Forest) to remove extreme values that deviated from the common pattern. This ensures that images with unusual dimensions do not skew the analysis. No anomalous images were found, confirming that all images met the expected criteria.

3.2.2 Handling Missing Values:

We ensured that no images had missing pixels or corrupt regions by checking for blank or black images. No images with missing pixels or corrupt areas were found, ensuring the dataset was clean and ready for further processing.

3.2.3 Dimensionality Reduction -

Image Resizing: To reduce computational load and maintain consistency, all images were resized to a standard size (224x224 pixels). This step helps streamline the model training process and ensures that all images are of a uniform size.

3.2.4 Data Balancing:

To ensure that the model is not biased toward one class, we analyzed the label distribution between "Bike" and "Car" categories. Although there was a slight imbalance, with 73 more car images than bike images (a 1.9% difference), the imbalance was minimal and acceptable for model training.

3.2.5 Normalization -

We applied pixel intensity normalization to scale pixel values to a [0,1] range. Additionally, we performed channel-wise normalization to standardize pixel values across color channels, improving model training by ensuring consistency in the data.

3.2.6 Feature Engineering

Feature engineering in this image dataset was minimal since deep learning models (CNNs) automatically extract spatial and texture features. However, essential preprocessing was performed to improve data quality. We removed 211 duplicate images using dHash, checked for anomalies with Isolation Forest, and confirmed no broken or corrupted images. All images were resized to 224x224 pixels for consistency and computational efficiency, with pixel values normalized to [0,1]. The dataset was already well-balanced, with only a 1.9% class imbalance, making further adjustments unnecessary. Since CNNs handle feature extraction, additional transformations were not required.

3.3 Model Selection-

3.3.1 Data Splitting

The dataset was split into training (70%), validation (15%), and test (15%) sets (2,649 - 568 - 568). This splitting method ensures that the model has sufficient data for training while also having distinct datasets to evaluate and test the model's performance.

3.3.2 Machine Learning Algorithms:

Logistic Regression

In the case of Logistic Regression, the images were first encoded into numerical labels and flattened into a 1D array to be processed by the model (pixels). The model was trained using the training set and tested on the validation set. LR models the probability of an image belonging to a certain class using a sigmoid function, producing class probabilities.

Random Forest

For Random Forest, the images were handled similarly by flattening the pixel data and converting it into numerical labels. The model was trained using a set of 100 decision trees, each trained on different subsets of the data. Random Forest aggregates predictions from all trees, enhancing its classification accuracy.

3.3.3 Neural Networks-

For the Neural Network model, a Convolutional Neural Network (CNN) was used to classify images. The architecture consists of three convolutional layers with ReLU activation and max-pooling, followed by two fully connected layers with dropout for regularization. The model was trained using the Adam optimizer and cross-entropy loss over three epochs.

3.3.4 Hyperparameter Tuning

Hyperparameter tuning was conducted to systematically analyze their impact by varying key parameters, including the learning rate, dropout rate, and weight decay. We trained multiple models with different configurations to assess their impact on the model performance. The learning rate was tested at three values [0.0001, 0.001, 0.01] to analyze its effect on convergence speed and stability. Dropout rates [0.3, 0.5, 0.7] were adjusted to evaluate their influence on overfitting, while weight decay values [0, 0.0005, 0.0001] were explored to regulate model complexity and prevent excessive parameter updates. Each configuration was trained for five epochs using the Adam optimizer and cross-entropy loss function. The tuning process highlighted the sensitivity of CNNs to hyperparameters, demonstrating that moderate dropout and a well-balanced learning rate contributed to improved generalization and classification accuracy.

3.3.5 Dataset Expansion through Augmentation

To enhance model robustness and improve classification performance, data augmentation techniques were applied to the dataset. The augmentation pipeline included horizontal flipping, vertical flipping, rotation, and color jittering, ensuring that the model learned from diverse image transformations. The dataset consisted of 3,649 images, with 1,988 bikes and 1,661 cars. The images were normalized and the model used for training was the same CNN described in Section 3.3.3.

3.3.6 Dataset Reduction via Undersampling

To evaluate model performance with a smaller dataset, undersampling was applied by randomly selecting 100 instances from the original training set. The selected subset maintained the original class distribution while significantly reducing the number of training samples. The same CNN described in Section 3.3.3 was trained using the Adam optimizer and cross-entropy loss over three epochs.

3.3.7 Improving the Architecture

To enhance performance, we introduced CNN2, an upgraded Convolutional Neural Network (CNN) with four convolutional layers using ReLU activation and max-pooling for deeper feature extraction. Compared to the original CNN, which had three convolutional layers and two fully connected layers, we added an additional convolutional layer and an extra fully connected layer to improve learning capacity. The fully connected layers were expanded to 1024 and 512 neurons, with dropout (0.5) applied to reduce overfitting. The model was trained with the Adam optimizer and cross-entropy loss for three epochs, with training time tracked to assess efficiency improvements.

4. RESULTS

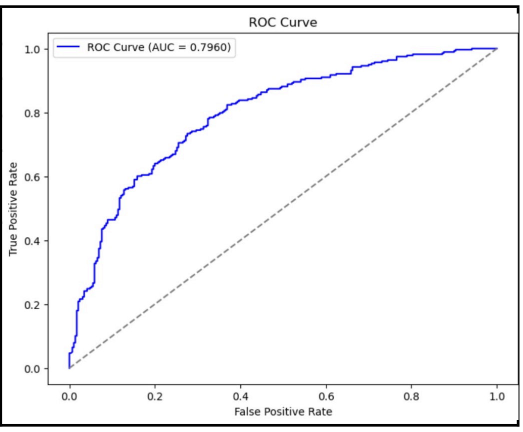
4.1 Machine Learning Algorithms:

Logistic Regression

The model achieves 100% training accuracy, indicating potential overfitting, while its validation accuracy is 72.01%, suggesting moderate generalization. With a recall of 80.22%, the model effectively identifies positive cases, but a precision of 68.2% indicates a notable false positive rate. The F1-score of 73.72% balances these metrics, and the ROC-AUC score of 0.7960 reflects reasonable class separation. The confusion matrix highlights a disparity in classification performance. While 223 bikes are correctly identified, 55 are misclassified as cars. In contrast, car classification is weaker, with 186 correctly classified but 104 misclassified as bikes. This suggests a bias toward predicting bikes, potentially due to class imbalance or overlapping features, which may be improved through data augmentation or model adjustments.

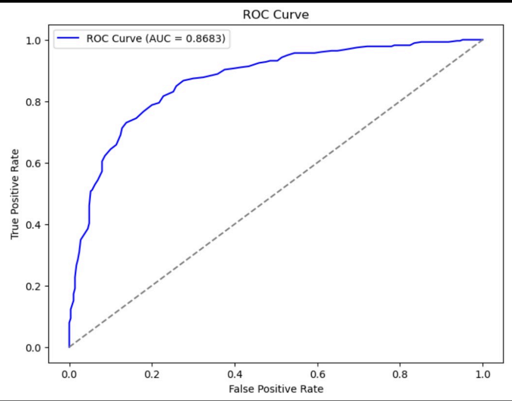
Training Accuracy	100%
Accuracy	72.01%
Recall	80.22%
Precision	68.2%
F1 Score	73.72%
ROC-AUC Score	0.7960

Confusion Matrix		
	Car	Bike
Car	186	104
Bike	55	223



Random Forest

The model achieves 100% training accuracy, indicating potential overfitting, with a validation accuracy of 79.4%, suggesting good generalization. It has a recall of 84.89% and precision of 75.88%, balancing false positives and negatives with an F1-score of 80.14%. The ROC-AUC score of 0.8683 reflects strong class separation. The confusion matrix shows 236 correctly classified bikes, with 42 misclassified as cars, while 215 cars are correctly identified, but 75 are misclassified as bikes. While overall performance is strong, the model struggles more with car classification.

Training Accuracy	100%		**Confusion Matrix**			
Accuracy	79.4%			Car	Bike	
Recall	84.89%		Car	215	75	
Precision	75.88%		Bike	42	236	
F1 Score	80.14%					
ROC-AUC Score	0.8683					

4.2 Neural Networks

The model achieves a training accuracy of 95.24%, indicating a strong fit to the training data, with a validation accuracy of 91.73%, suggesting great generalization. It maintains a recall of 92%, meaning it effectively identifies most positive cases, and a precision of 92%, indicating a low false positive rate. The F1-score of 91.97% reflects a well-balanced trade-off between precision and recall. The confusion matrix shows strong classification performance, correctly identifying 269 bikes, with only 9 misclassified as cars. Similarly, 252 cars are correctly classified, while 38 are misclassified as bikes. These results suggest that the model is highly effective in distinguishing between the two classes, with minimal misclassification.

Training Accuracy	95.24%		**Confusion Matrix**		
Accuracy	91.73%				
Recall	92%			Car	Bike
Precision	92%		Car	252	38
F1 Score	91.97%		Bike	9	269

4.3 Hyperparameter Tuning

Learning Rates:

The model's performance varies significantly across different learning rates (LR). At learning rate 0.0001 the model achieves the most balanced performance. Achieving a training accuracy of 97.24% and a validation accuracy of 92.43%, with a recall of 93.17% and a precision of 91.52%, leading to an F1-score of 92.34%. The confusion matrix indicates strong classification accuracy, correctly identifying 266 cars and 259 bikes, with minimal misclassification. When the learning rate was increased to 0.001, training accuracy improved slightly (98.72%), but validation accuracy remained similar (92.25%), indicating marginal improvement. The model identified 260 cars and 264 bikes correctly, with 30 cars and 14 bikes misclassified. However, a significantly higher learning rate of 0.01 resulted in severe performance degradation, with the model failing to generalize (training accuracy: 51.72%, validation accuracy: 48.94%). The confusion matrix for LR = 0.01 shows complete misclassification, where all cars were classified as bikes and vice versa, highlighting the detrimental impact of an excessively high learning rate.

	LR=0.0001			**LR=0.001**			**LR=0.01**		
Training Accuracy	97.24%			98.72%			51.72%		
Accuracy	92.43%			92.25%			48.94%		
Recall	93.17%			94.96%			100%		
Precision	91.52%			89.80%			48.94%		
F1 Score	92.34%			92.31%			65.72%		
Confusion Matrix	**Confusion Matrix**			**Confusion Matrix**			**Confusion Matrix**		
		Car	Bike		Car	Bike		Car	Bike
	Car	266	24	Car	260	30	Car	0	290
	Bike	19	259	Bike	14	264	Bike	0	278

Dropout Rates:

The results indicate that increasing dropout generally improves generalization. At dropout =

0.3, the model achieved 91.02% accuracy, with a relatively lower recall (87.77%) but high precision (93.49%), suggesting a stronger emphasis on minimizing false positives. With dropout = 0.5, recall increased significantly (94.6%), but precision dropped to 87.38%, indicating more aggressive learning. The highest dropout rate (0.7) resulted in the best overall accuracy (93.49%), with both high recall (96.4%) and strong precision (90.85%), leading to the highest F1-score of 93.54%. The confusion matrices confirm that higher dropout values improve recall by reducing false negatives, though at the cost of slightly lower precision.

	Dropout = 0.3			**Dropout = 0.5**			**Dropout = 0.7**		
Training Accuracy	96.79%			99.74%			98.79%		
Accuracy	91.02%			90.67%			93.49%		
Recall	87.77%			94.6%			96.4%		
Precision	93.49%			87.38%			90.85%		
F1 Score	90.54%			90.85%			93.54%		
Confusion Matrix	**Confusion Matrix**			**Confusion Matrix**			**Confusion Matrix**		
		Car	Bike		Car	Bike		Car	Bike
	Car	273	17	Car	252	38	Car	263	27
	Bike	34	244	Bike	15	263	Bike	10	268

Weight Decay:

With no weight decay (0), the model achieved 99.06% training accuracy, indicating potential overfitting, though validation accuracy remained high at 92.96%. Recall and precision were well-balanced (93.88% and 91.90%, respectively), leading to an F1-score of 92.88%. Introducing weight decay = 0.0005 reduced training accuracy (97.89%) but also slightly decreased overall validation performance (91.02% accuracy), with higher recall (94.96%) but lower precision (87.71%), suggesting a shift towards greater sensitivity at the cost of more false positives. The best overall performance was observed with weight decay = 0.0001, achieving 93.13% accuracy, 94.96% recall, and 91.35% precision, leading to the highest F1-score of 93.12%. The confusion matrices confirm that weight decay helps balance false positives and false negatives.

	weight decay 0			**weight decay 0.0005**			**weight decay 0.0001**		
Training Accuracy	99.06%			97.89%			98.64%		
Accuracy	92.96%			91.02%			93.13%		
Recall	93.88%			94.96%			94.96%		
Precision	91.90%			87.71%			91.35%		
F1 Score	92.88%			91.19%			93.12%		
Confusion Matrix	**Confusion Matrix**			**Confusion Matrix**			**Confusion Matrix**		
		Car	Bike		Car	Bike		Car	Bike
	Car	267	23	Car	253	37	Car	265	25
	Bike	17	261	Bike	14	264	Bike	14	264

4.4 Dataset Expansion through Augmentation

The model achieves a training accuracy of 95.01%, indicating a strong fit to the training data, with a validation accuracy of 95.42%, suggesting good generalization. It maintains a recall of 95%, meaning it effectively identifies most positive cases, and a precision of 96%, indicating a low false positive rate. The F1-score of 95.2% reflects a well-balanced trade-off between precision and recall. The confusion matrix shows strong classification performance, correctly identifying 258 bikes, with only 6 misclassified as cars. Similarly, 284 cars are correctly classified, while 20 are misclassified as bikes. These results suggest that the model is highly effective in distinguishing between the two classes, with minimal misclassification.

Training Accuracy	95.01%		**Confusion Matrix**		
Accuracy	95.42%				
Recall	95%			Car	Bike
Precision	96%		Car	284	6

F1 Score	95.2%		Bike	20	258

4.5 Dataset Reduction via Undersampling

The model trained on the undersampled dataset achieved a training accuracy of 54% and a validation accuracy of 48.94%, indicating poor generalization. The recall (49%) and precision (24%) were significantly lower than in other models, leading to an F1-score of 65.72%. The confusion matrix shows that all 290 cars were misclassified as bikes, and all 278 bikes were also misclassified as cars, demonstrating that reducing the dataset severely impacted classification performance.

Training Accuracy	54%		**Confusion Matrix**		
Accuracy	48.94%				
Recall	49%			Car	Bike
Precision	24%		Car	0	290
F1 Score	65.72%		Bike	0	278

4.6 Improving the Architecture

The improved CNN2 architecture achieved a training accuracy of 93.24% and a validation accuracy of 93.84%, indicating strong performance. The model maintained a recall and precision of 94%, leading to an F1-score of 93.69%. The confusion matrix shows 273 correctly classified cars and 260 correctly classified bikes, with 17 cars and 18 bikes misclassified. The total training time was 270 seconds, reflecting the computational cost of the deeper architecture.

Training Accuracy	93.24%		**Confusion Matrix**		
Accuracy	93.84%				
Recall	94%			Car	Bike
Precision	94%		Car	273	17
F1 Score	93.69%		Bike	18	260
Training Time	270s				

4.7 Performance on the test dataset

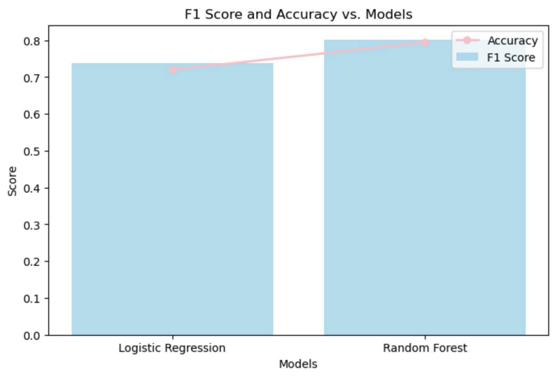
After conducting various experiments on the validation set, we will now evaluate the performance on the test set using the insights we gained. We will utilize the CNN architecture, as it delivered the best results. Specifically, we will use the improved CNN2 model, which achieved better validation performance. The model will be trained on the augmented data, and we will apply the optimal hyperparameters discovered during our experiments. While we acknowledge that the hyperparameters might differ for CNN2 and the augmented data, for the purposes of this project, using the same hyperparameters is sufficient.

Training Accuracy	90.96%		**Confusion Matrix**		
Accuracy	93.66%				
Recall	94%			Car	Bike
Precision	94%		Car	277	10
F1 Score	93.41%		Bike	26	255

5. DISCUSSION AND CONCLUSIONS

5.1 Comparison of Logistic Regression and Random Forest Performance

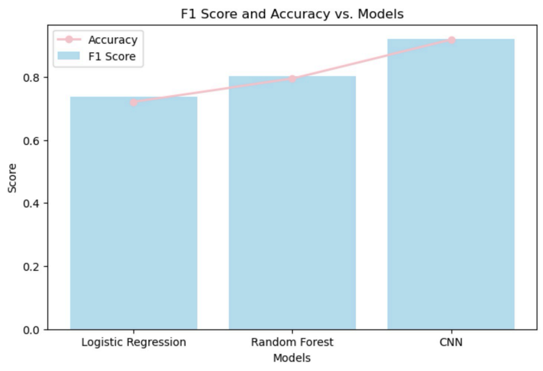
The Random Forest model outperforms Logistic Regression in all key metrics, achieving higher accuracy (79.4% vs. 72.01%), recall (84.89% vs. 80.22%), precision (75.88% vs. 68.2%), and ROC-AUC (0.8683 vs. 0.7960). Additionally, RF demonstrates better generalization, as both models achieve 100% training accuracy, but RF maintains higher validation performance. RF's decision tree-based structure enables it to better capture complex patterns, leading to improved classification performance, particularly in distinguishing bikes, with fewer misclassifications (42 vs. 55). Logistic Regression, being a linear model, struggles with non-linear decision boundaries, making it less effective in handling intricate relationships within the data. However, Logistic Regression remains simpler, computationally efficient, and less prone to overfitting, while Random Forest, despite offering better predictive performance, may require more fine-tuning to optimize its results.



5.2 Comparison of Machine Learning and Neural Network Performance

Neural Networks outperform traditional Machine Learning models across all key metrics. The NN model achieves the best training accuracy (95.24%) and validation accuracy (91.73%), compared to 100% training accuracy but lower validation accuracy in Random Forest (79.4%) and Logistic Regression (72.01%), indicating better generalization. NN also excels in recall (92%) and precision (92%), resulting in an F1-score of 91.97%, significantly reducing misclassification rates. The confusion matrices show that NN correctly classifies 269 bikes and 252 cars, misclassifying only 9 bikes and 38 cars, far fewer errors than both ML models.

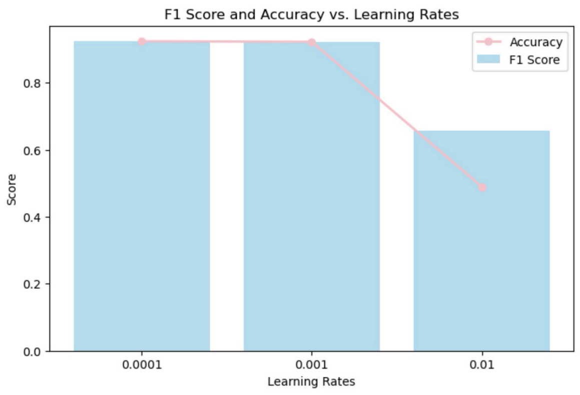
Scientifically, CNN's advantage lies in its ability to extract hierarchical spatial features from image data, unlike ML models, which rely on hand-crafted features or simple decision boundaries. While Random Forest improves upon Logistic Regression by leveraging ensemble learning, it still struggles with complex feature extraction. CNN, however, learns intricate patterns automatically, leading to more accurate classifications. Despite these advantages, deep learning requires greater computational resources and training time, whereas ML models remain more interpretable and computationally efficient. Nonetheless, when accuracy and classification performance are the primary concerns, CNN provides the best decision boundaries and generalization, making it the superior choice for this task.



5.3 Hyperparameter Comparison

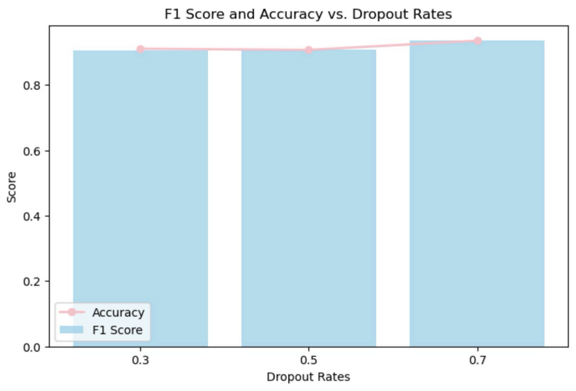
Learning Rates:

The learning rate plays a crucial role in model convergence and generalization. A low learning rate (0.0001) allows for gradual updates, preventing overshooting of the loss function's optimal point, leading to high accuracy and a well-balanced precision-recall tradeoff. The results indicate that 0.0001 achieved the best generalization with minimal misclassification. Increasing the learning rate to 0.001 led to slightly improved recall (94.96%) but lower precision (89.80%), suggesting that the model became more sensitive to positive cases but at the expense of an increased false positive rate. This trade-off highlights how a moderate learning rate can impact classification performance by prioritizing sensitivity over specificity. However, with LR = 0.01, the model exhibited clear signs of instability and divergence, failing to learn meaningful patterns. The drastic drop in accuracy (48.94%) suggests that weight updates were too large, preventing the model from settling into an optimal solution. The confusion matrix further confirms this, as all samples were classified into a single category, indicating that the model did not successfully differentiate between classes. These findings emphasize the importance of choosing an optimal learning rate to ensure both convergence and high classification performance.



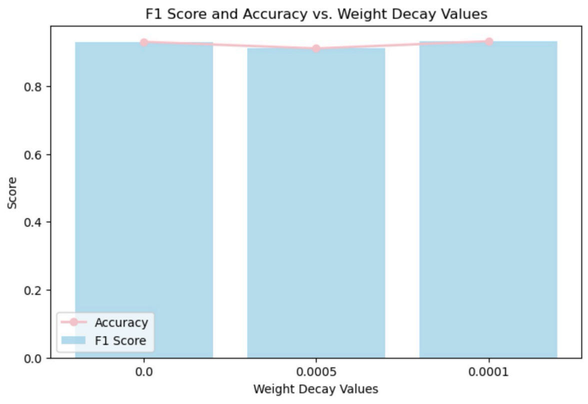
Dropout Rates:

Dropout is a crucial regularization technique that helps prevent overfitting by randomly deactivating neurons during training. The results demonstrate that increasing dropout enhances recall, as seen in the jump from 87.77% (dropout = 0.3) to 96.4% (dropout = 0.7), meaning the model becomes more effective at identifying true positives. However, precision declines slightly at higher dropout values, indicating an increase in false positives. The dropout = 0.7 configuration achieved the best balance, with the highest F1-score (93.54%), showing that a moderate-to-high dropout rate improves generalization without sacrificing classification performance. Conversely, at dropout = 0.3, while precision was higher, recall was significantly lower, implying the model was more conservative in identifying positive cases. These findings highlight that a higher dropout rate leads to better generalization and robustness, but an optimal balance must be maintained to avoid excessive loss of useful information.



Weight Decay Values:

Weight decay acts as an L2 regularization technique to prevent overfitting by penalizing large weights. The results show that applying weight decay reduces overfitting, as seen in the lower training accuracy while maintaining high validation accuracy. Without weight decay (0), the model had the highest training accuracy (99.06%), suggesting it learned the training data too well, which can lead to poor generalization. By introducing weight decay at 0.0005, recall improved, but precision dropped, indicating the model became more aggressive in classifying positive instances at the cost of more false positives. The optimal balance was found at weight decay = 0.0001, where both recall (94.96%) and precision (91.35%) were strong, leading to the highest F1-score (93.12%). This suggests that a small amount of weight decay enhances generalization, whereas too little allows overfitting, and too much may limit the model’s capacity to learn complex patterns.



5.4 Comparison of Dataset Modification Techniques

The results demonstrate that dataset augmentation significantly enhances model performance, while undersampling severely impairs classification accuracy and generalization. Augmentation improved validation accuracy from 91.73% to 95.42% and the F1-score from 91.97% to 95.2%, indicating that exposure to diverse transformations reduced overfitting and improved robustness. The confusion matrix confirms this, showing a decrease in misclassified cars from 38 to just 6, suggesting that augmentation helps the model learn more invariant representations of the data. In contrast, undersampling drastically reduced validation accuracy to 48.94%, with the model failing to classify any cars correctly, instead misclassifying all 290 cars as bikes. This highlights a key limitation of undersampling: while it may help address class imbalance, excessive reduction in training data hinders the model's ability to learn meaningful patterns, leading to extreme misclassification errors. These findings reinforce that data augmentation is a highly effective strategy for improving model generalization, whereas undersampling must be applied carefully to avoid degrading classification performance.

	Original Dataset			**Dataset Expansion through Augmentation**			**Dataset Reduction via Undersampling**		
Training Accuracy	95.24%			95.01%			54%		
Accuracy	91.73%			95.42%			48.94%		
Recall	92%			95%			49%		
Precision	92%			96%			24%		
F1 Score	91.97%			95.2%			65.72%		
Confusion Matrix	**Confusion Matrix**			**Confusion Matrix**			**Confusion Matrix**		
		Car	Bike		Car	Bike		Car	Bike
	Car	252	38	Car	284	6	Car	0	290
	Bike	9	269	Bike	20	258	Bike	0	278

5.5 Comparison of Original and Improved Architecture

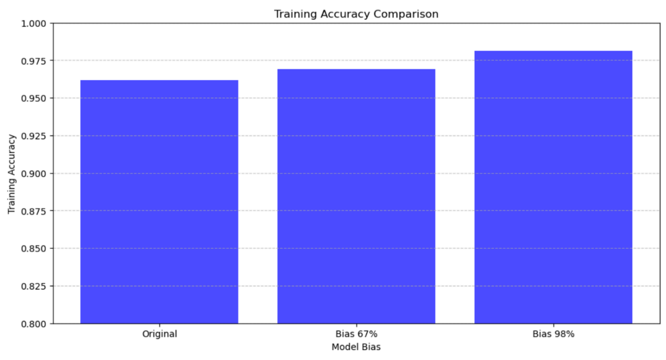
The improved CNN2 architecture demonstrated enhanced classification performance compared to the original CNN model. The validation accuracy increased from 91.73% to 93.84%, and the F1-score improved from 91.97% to 93.69%, indicating better overall generalization. Additionally, recall and precision both increased from 92% to 94%, highlighting the model's improved ability to correctly classify instances. The confusion matrix reveals a significant reduction in misclassifications. The original model misclassified 38 cars and 9 bikes, while the improved model reduced this to 17 cars and 18 bikes, improving classification consistency. However, this performance gain came at a computational cost, as the training time increased from 240s to 270s, reflecting the additional complexity of the deeper network. These results suggest that adding more convolutional layers and dropout improved classification accuracy and robustness, albeit with a slight increase in training time.

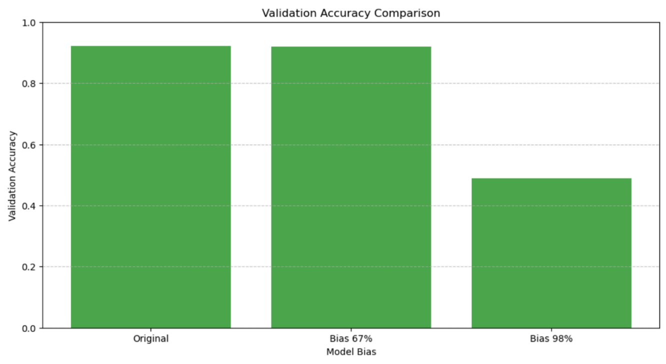
	Original Architecture			**Improved Architecture**		
Training Accuracy	95.24%			93.24%		
Accuracy	91.73%			93.84%		
Recall	92%			94%		
Precision	92%			94%		
F1 Score	91.97%			93.69%		
Training Time	240s			270s		
Confusion Matrix	**Confusion Matrix**			**Confusion Matrix**		
		Car	Bike		Car	Bike
	Car	252	38	Car	273	17
	Bike	9	269	Bike	18	260

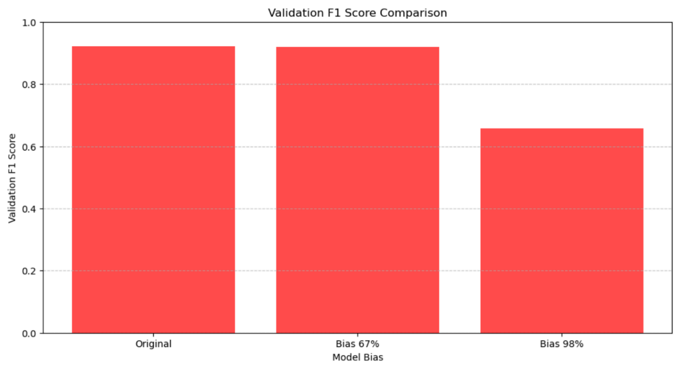
5.6 Adding a new metric for training evaluation

We chose the ROC AUC (Receiver Operating Characteristic Area Under the Curve) metric because it evaluates the model's ability to distinguish between the positive and negative classes across different thresholds, providing a comprehensive measure of performance. This metric is particularly useful for binary classification tasks, as it captures both the true positive rate (sensitivity) and false positive rate, offering a clear view of the model's classification ability regardless of class imbalance. A higher AUC score indicates better model performance in distinguishing between the two classes. We didn't use it in the deep learning model yet, and we will add the softmax activation for calculating probabilities in order to do so.
The training ROC curves show significant improvement in the model's performance over the course of three epochs. In the first epoch, the AUC was 0.80, indicating a relatively weak ability to distinguish between the classes. By the second epoch, the AUC increased to 0.97, demonstrating considerable improvement. In the third epoch, the model achieved an AUC of 0.99, reflecting strong discriminatory ability between the classes. The training accuracy also increased to 96.19%, highlighting the model's successful learning and convergence. These results indicate that the model's performance improved steadily with training (faster in the first epochs), becoming highly effective by the end of the third epoch.

5.7 The Effect of Data Imbalance on Model Performance







We modified the class distribution of the train dataset to test how different levels of class imbalance impact the model's performance. The original dataset was balanced, with both classes having an almost equal representation (50%). In the second scenario, we created a bias by adjusting class 1 to be 67% of the dataset. This led to a slight change in performance, with training accuracy improving to 96.90%, but validation accuracy decreased to 91.90%. In contrast, the F1 score remained relatively similar (0.92).
In the third scenario, we further increased the class imbalance by adjusting class 1 to account for 98% of the data, resulting in a severe class imbalance. The model's training accuracy increased to 98.14%, but the validation accuracy dramatically dropped to 48.94%, and the F1 score also decreased to 0.66. The high bias caused the model to predict exclusively class 1, as it learned to focus on the majority class. This stark contrast between training and validation accuracy shows how excessive imbalance can lead to overfitting, where the model performs well on the training set but poorly on unseen data.

From this analysis, it is evident that while increasing class bias can improve training accuracy, it significantly harms the model's generalization ability, as seen in the drop in validation metrics. Maintaining a balanced dataset or applying techniques to handle class imbalance is crucial for ensuring robust model performance.

5.8 Dimensionality Reduction and Its Impact on Classification

We employed two different dimensionality reduction methods:

PCA:

This technique requires flattening the data and discards spatial relationships, making it suitable for use with Random Forest.

To reduce the dimensionality of the dataset, we applied Principal Component Analysis (PCA), which is a widely used technique for retaining most of the variance while reducing the number of features. After flattening the image data, we used PCA to reduce the feature space to 50 components, from 224x224x3 which is 150K components (only 0.03%). This dimensionality reduction helped improve the model's performance by reducing overfitting and improving generalization. After applying PCA, the validation accuracy increased from 79.05% to 81.34%, and the F1 score, precision, and recall also improved. In conclusion, PCA effectively improved the performance and efficiency of the Random Forest model by reducing the feature space while retaining important information.

Max Pooling:

This method preserves the spatial relationships in the data, making it ideal for use with Convolutional Neural Networks (CNNs).

We used MaxPooling before the CNN layers as a dimensionality reduction method. By applying MaxPooling first, we reduced the spatial dimensions of the input data before feeding it into the convolutional layers. This technique reduces the amount of computational power required by decreasing the size of the data passed through the model.

The training time was significantly shorter, totaling 49.24 seconds for 3 epochs (instead 240 seconds). This is because the data passed to the convolutional layers was smaller, which reduced the overall computational load. Validation accuracy was slightly lower compared to the baseline model, dropping from over 0.92 to 0.9102. This is expected, as reducing the dimensions before convolution might lead to a loss of spatial information, potentially impacting the model's ability to fully capture complex features. F1 score was quite high at 0.9140, showing that despite the reduction in dimensionality, the model still performed well in terms of both precision and recall. In conclusion, MaxPooling before the convolutional layers provided a trade-off between computational efficiency (lower training time) and a minor decrease in accuracy.

6. FUTURE WORK

While this study successfully improved image classification performance, several areas remain for future exploration. First, extending the dataset with more diverse samples could enhance generalization. Additionally, experimenting with deeper architectures, such as ResNet or Vision Transformers, may further improve accuracy. Transfer learning using pre-trained models could also be beneficial, especially for small datasets. Another avenue for research is optimizing hyperparameters with automated search techniques, such as Bayesian optimization or reinforcement learning. Lastly, integrating additional metrics and explainability techniques (e.g., Grad-CAM) could provide deeper insights into model performance.

7. REFERENCES

Frisk, M., Storgaard, K., & Gottfredsen, J. P. (2023). MULTI PRETEXT TASK SELF-SUPERVISED LEARNING FOR CULTURAL HERITAGE CLASSIFICATION. Aarhus University, Department of Computer Science.