Exploring the Impact of Image Permutation on Classification Performance

Shelly Levy 318901550 and Orian Ganor 318862224

Dreamed big, had low computational power, but we made it work! Thanks for the adventure! 🚀🧠💻

ABSTRACT

This project delves into the intriguing realm of image permutation and its influence on the performance of neural network classifiers. By segmenting images into tiles and applying various permutation patterns, we aim to unravel the intricate dynamics of how neural networks perceive and process altered spatial relationships. The granularity of tiling, the complexity of permutation patterns, and the depth of network architectures are meticulously examined to provide comprehensive insights into their effects on classification accuracy. Additionally, the project explores the potential of pretrained convolutional neural networks, the impact of image content, the efficacy of augmentation techniques, and the interpretability of learned features in the context of permuted images. Through rigorous experimentation and analysis, we seek to understand how tile resolution and permutation patterns influence the feasibility of classification and the learning process of neural networks.

1. INTRODUCTION

The robustness of neural network models, particularly in the realm of image classification, is a critical area that addresses the models' ability to maintain high performance under various image permutations. This project focuses on one such permutation: the permutation of image tiles. By dissecting images into smaller segments and reorganizing these segments, we can challenge neural networks in unique ways, unveiling their capabilities and limitations in recognizing and adapting to altered spatial relationships.

Our investigation delves into several core aspects of image permutation's impact on neural network performance. We explore the granularity of image tiling, positing that there exists an optimal level of granularity where network performance is maximized, balancing detail preservation and the risk of overfitting. The complexity of permutation patterns is examined with the hypothesis that increased complexity may impair the network's feature extraction capabilities, thus deteriorating classification performance. We also consider the depth and complexity of neural network architectures, suggesting that deeper networks might better adapt to permutations, albeit with potential diminishing returns or overfitting risks beyond a certain depth.

The project further compares the effectiveness of pretrained versus custom-designed convolutional neural networks (CNNs) in adapting to permuted images, hypothesizing that pretrained CNNs, with their extensive prior learning, might have an edge in performance. The role of image augmentation techniques, particularly those introducing spatial variability, is assessed for their potential to enhance the classification of permuted images.

Additionally, we examine the interpretability of learned features post-permutation, the differential impact of various permutation types on classification accuracy, and how spatial-aware augmentation techniques might help neural networks better generalize to permuted images. The effect of hyperparameter tuning on the network's performance on permuted image datasets is also a key focus, exploring how adjustments in batch normalization, batch size, epoch size and learning rate can fine-tune

the network's adaptability and robustness in the face of spatial permutations in the input data. Through these explorations, the project aims to provide a nuanced understanding of how image permutation influences classification performance and to identify strategies to enhance neural network robustness in this context.

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:

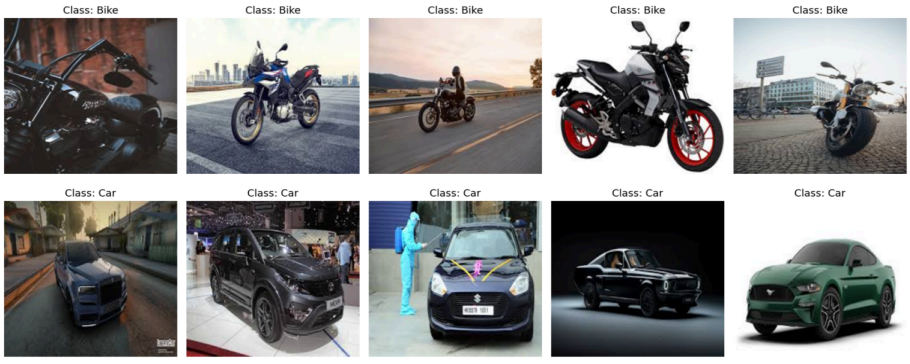
Dataset and Data Augmentation: The research utilized augmentation techniques such as rotation, jigsaw puzzle permutation, and image recreation from inpainted images to expand the training dataset. This approach enhanced the model's performance. We plan to employ similar augmentation strategies to increase our dataset, potentially improving our models's robustness and accuracy.

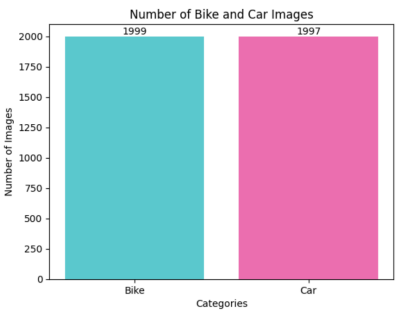
CNN Architecture: Various CNN architectures were explored in the research, including a baseline model, a deep bottleneck model inspired by ResNet50, and a transfer learning model using ResNet50. The superior performance of the transfer learning model indicates that we should consider this approach. Also 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. Dataset Description

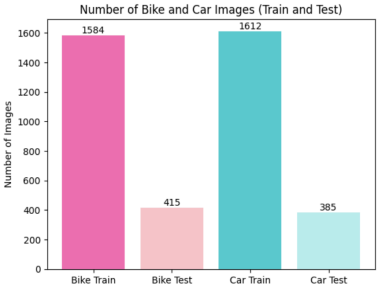
We have selected a dataset consisting of images depicting bikes and cars, with the objective of training a model to accurately classify these images. The dataset comprises approximately 3996 images, and the distribution between the two categories is relatively uniform.





3.1 Baseline

The baseline dataset comprises the original images, against which we can evaluate the performance of various model permutations. We will partition it into training and testing sets using a random state to ensure uniform distribution across all datasets we create subsequently. We can observe that the distribution between cars and bikes remains relatively consistent between the training and testing sets.

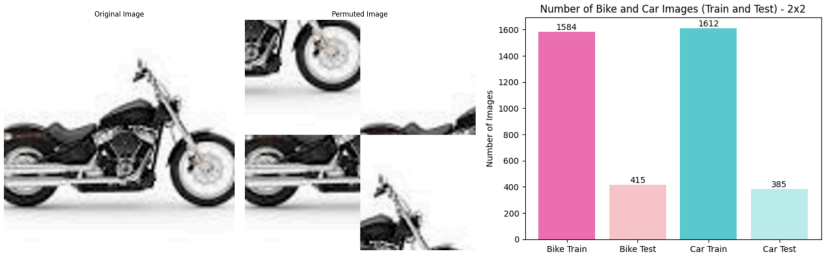


3.2 Permutations

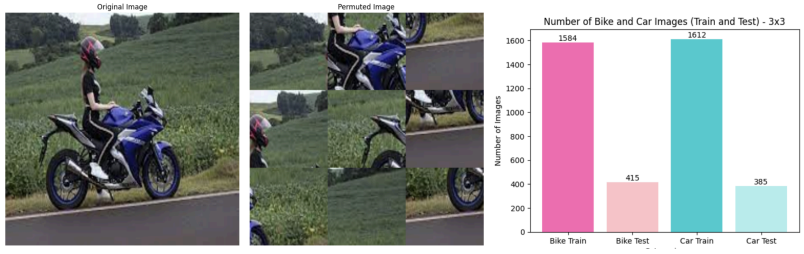
We generated random permutations using different numbers of tiles (2, 3, 4, 20). We selected these numbers to observe transformations ranging from what we consider easy to more complex variations. Each of these datasets was saved on the computer and loaded for each rerun to maintain consistent permutations across runs.

We split the dataset into training and testing subsets while maintaining the same random state to ensure consistency across datasets for accurate comparison. It should be noted that the distribution of bicycles and cars remains relatively consistent between the training and testing sets.

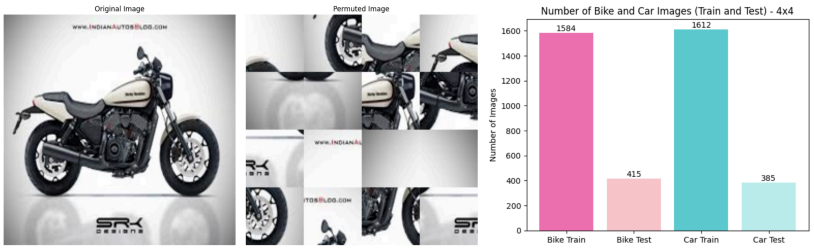
2-tile permutations-



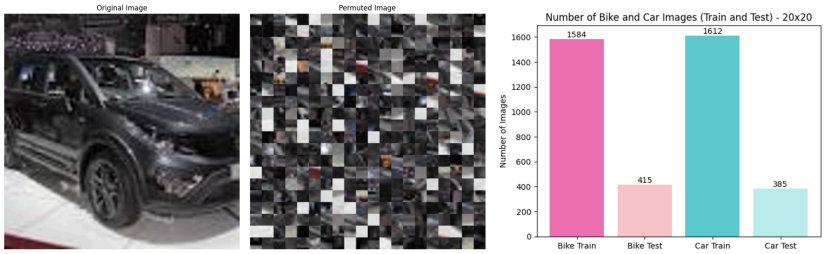
3-tile permutations-



4-tile permutations-



20-tile permutations-



4. Improving Performance and Tile Resolution Impact-

The forthcoming analysis delves into the impact of endeavors to enhance network performance within the converted environment, alongside an examination of the influence of tile resolution on the feasibility of classification subsequent to conversion.

4.1 Impact of Architectural Variations-

In our investigation into the impact of image permutation on classification performance, we employed a diverse array of neural network architectures, scrutinizing their adaptability and resilience with a focus on depth, complexity, architectural features, and pretraining influences. We explored four architectures to ascertain how each responds to the challenges posed by permuted images.

- ResNet-18: This architecture incorporates skip connections, a feature that allows it to effectively handle the vanishing gradient problem by enabling gradient flow directly across layers. With 18 layers, ResNet-18 can strike an efficient balance between depth and computational demand, making it adept at adapting to various tasks, especially with its pretraining on the ImageNet dataset which enhances its feature extraction and transfer learning capabilities.

- GoogleNet: GoogleNet's core feature is its inception modules, which combine convolutional layers of different sizes to capture both detailed and broad features within an image. This 22-layer network is designed for efficiency and depth, incorporating auxiliary classifiers to enhance training by providing additional feedback on the network's performance at intermediate layers. Pretraining on ImageNet can boost its capability to extract and leverage complex features for varied image recognition tasks.

- DenseNet-121: DenseNet-121 features a unique dense connectivity pattern, connecting each of its 121 layers to every other layer, which enhances feature propagation and reuse throughout the network. This architecture not only can improve parameter efficiency but also combats the vanishing gradient problem. Pretraining on ImageNet can further capitalize on these architectural benefits, enhancing the network's effectiveness in transfer learning and across various image recognition tasks.

- Custom CNN: This network structure starts with five convolutional layers, incrementally increasing channel depth, interspersed with ReLU activations and selective max pooling to reduce spatial dimensions, while the initial layers preserve spatial detail by omitting pooling. The architecture transitions from feature extraction to classification through three dense layers, culminating in a binary output. Trained from scratch, allowing it to be specifically tailored to the dataset without relying on pretraining.

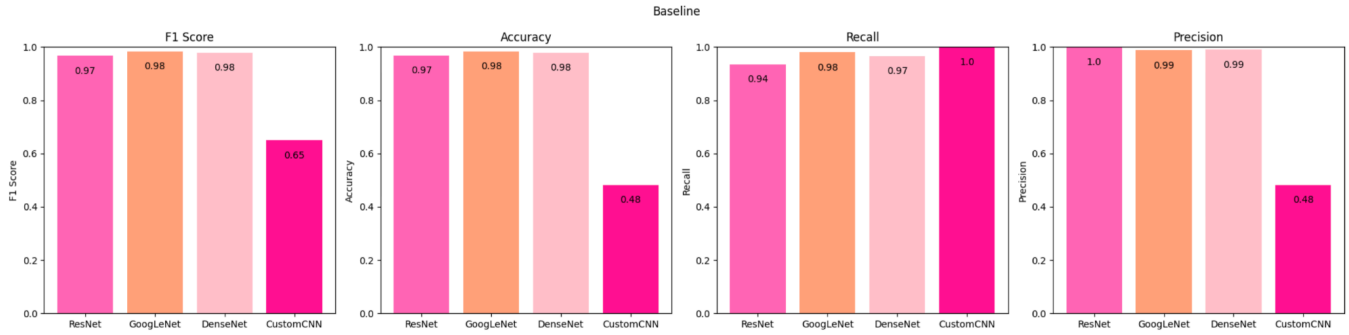
4.1.1 Architectural Performance Comparison Across Different Permutations

ResNet-18, GoogleNet, and DenseNet-121 showcase the highest F1 scores and accuracy, which seems to be a testament to the advantages of deep architectures that incorporate sophisticated mechanisms for managing depth. It look that ResNet-18's skip connections help mitigate the vanishing gradient problem, allowing it to learn effectively despite its depth, GoogleNet's modular approach with inception modules enables it to capture features at various scales efficiently, and DenseNet-121's feature reusability and reduced vanishing gradient effect due to its dense connectivity both likely contribute to their high scores. These factors, coupled with pretraining on extensive datasets like ImageNet, enhance their feature extraction capabilities and generalization to the dataset used in this project.

In contrast, the Custom CNN exhibits a significantly lower F1 score and accuracy. The absence of pre-training means that it starts learning from scratch, which may not offer robust feature extraction capabilities akin to pre-trained models, resulting in diminished performance. Regarding recall and precision, the Custom CNN achieves perfect recall, indicating its ability to correctly identify all relevant instances of the class. However, its accuracy is extremely low, suggesting a high rate of false positives.

This could imply that the network did not learn but merely made guesses. (reminder: we tested together and did not find any issues)

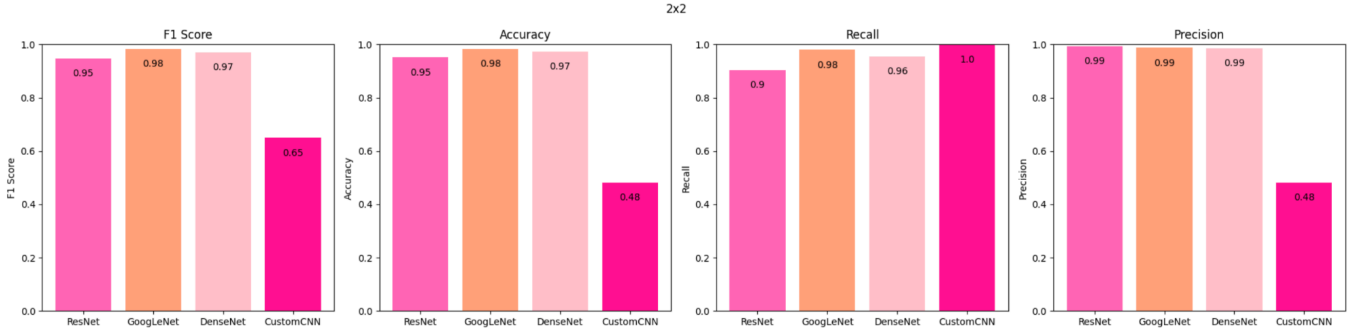
Table 1. Baseline Classification Performance Metrics for Neural Network Architectures				
	ResNet-18	GoogleNet	DenseNet-121	Custom CNN
Accuracy	0.97	0.98	0.98	0.48
F1 Score	0.97	0.98	0.98	0.65
Recall	0.94	0.98	0.97	1
Precision	1	0.99	0.99	0.48



In the 2-tile permutations scenario, ResNet-18, GoogleNet, and DenseNet-121 maintain high classification performance, with F1 scores above 0.95 and precision rates at or near 0.99. This indicates that even with image segments reorganized, the depth and sophisticated structures of these models are robust. ResNet-18's residual connections might be allowing it to preserve information across permuted segments, while GoogleNet's inception modules and DenseNet's dense connectivity likely aid in reassembling fragmented spatial information.

The Custom CNN, again while maintaining perfect recall, suffers in precision (0.48), suggesting that the network did not learn but merely made guesses. The absence of pretraining might hinder its initial feature extraction capacity, which is not as readily compensated for in the permuted context as in the architectures with more advanced features and depth.

Table 2. 2-Tile Permutations Classification Performance Metrics for Neural Network Architectures				
	ResNet-18	GoogleNet	DenseNet-121	Custom CNN
Accuracy	0.95	0.98	0.97	0.48
F1 Score	0.95	0.98	0.97	0.65
Recall	0.9	0.98	0.96	1
Precision	0.99	0.99	0.99	0.48



In the evaluation of the 3-tile permutations classification performance metrics, it's apparent that architectures with deeper layers and more sophisticated features, such as GoogleNet and DenseNet-121, maintain high performance levels. GoogleNet, with its inception modules and ability to capture multi-scale information, sustains a high accuracy (0.97) and F1 score (0.97), indicating its robust feature extraction even under permutation. Similarly, DenseNet-121 also shows resilience with an accuracy of

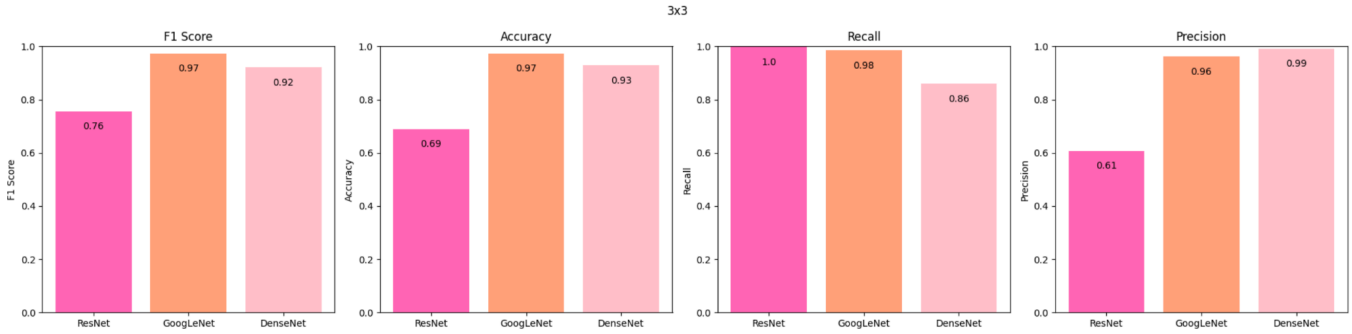
0.93 and F1 score of 0.92, likely due to its dense connectivity which might help in preserving information across the network even when spatial continuity is disrupted.

On the contrary, ResNet-18, although pre-trained and equipped with skip connections, sees a decline in performance (accuracy of 0.69, F1 score of 0.76), potentially due to the fewer layers compared to DenseNet-121 which might limit its ability to adapt to the permuted inputs.

The perfect recall scores of ResNet-18 suggest it is highly sensitive to the positive class, potentially at the cost of precision, leading to a higher rate of false positives. Conversely, the high precision of DenseNet-121 (0.99) indicates its effectiveness in making correct positive predictions, which could be due to the comprehensive feature retention facilitated by its architecture.

Table 3. 3-Tile Permutations Classification Performance Metrics by Architectures

	ResNet-18	GoogleNet	DenseNet-121
Accuracy	0.69	0.97	0.93
F1 Score	0.76	0.97	0.92
Recall	1	0.98	0.86
Precision	0.61	0.96	0.99

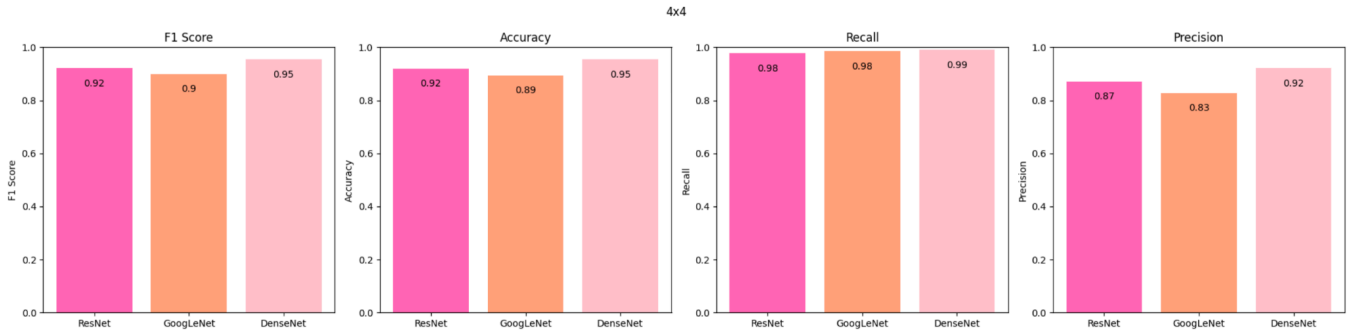


For the 4-tile permutations, ResNet-18 shows commendable resilience with an accuracy of 0.92 and F1 score of 0.92, likely due to its skip connections that may help preserve spatial hierarchies despite permutation. GoogleNet's performance, with an accuracy of 0.89 and F1 score of 0.9, underlines the robustness provided by its inception modules, which may allow for better recognition of features despite their displacement. DenseNet-121 maintains high performance metrics (accuracy of 0.95 and F1 score of 0.95), which can be ascribed to its dense connectivity pattern ensuring maximum information flow between layers for optimal feature reuse and recognition.

The high recall of ResNet-18, GoogleNet, and DenseNet-121 suggests they can correctly identify the majority of the positive class even after permutations. However, the drop in precision for ResNet-18 indicates a rise in false positives, which may result from the challenge of interpreting reorganized features. In contrast, DenseNet's relatively consistent precision suggests its architecture copes well with permutation, possibly through effective integration of features across its densely connected layers.

Table 4. 4-Tile Permutations Classification Performance Metrics by Architectures

	ResNet-18	GoogleNet	DenseNet-121
Accuracy	0.92	0.89	0.95
F1 Score	0.92	0.9	0.95
Recall	0.98	0.98	0.99
Precision	0.87	0.83	0.92



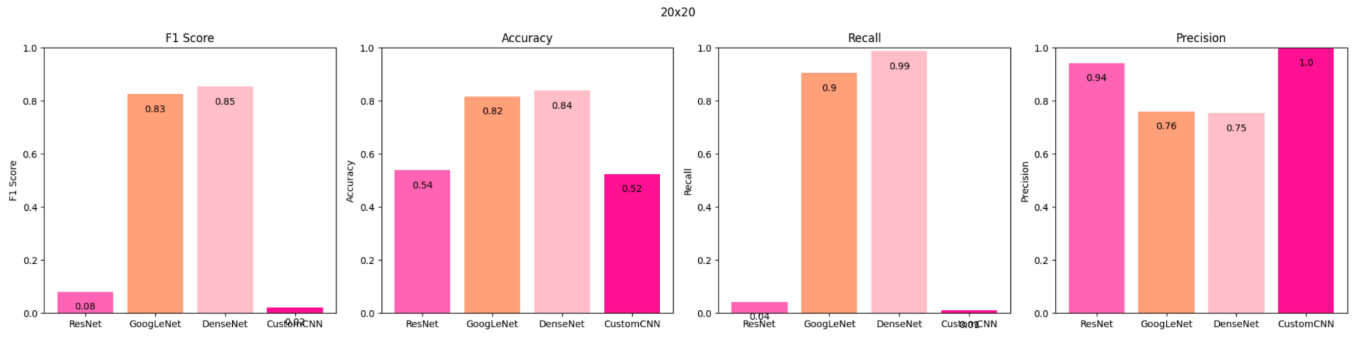
For the 20-tile permutations, ResNet-18 architecture, despite its depth and skip connections that generally contribute to better feature propagation, struggles with accuracy (0.54) and particularly F1 score (0.08), indicating difficulties in handling highly granular permutations. The high precision (0.94) but very low recall (0.04) suggests that while it's highly confident about the few predictions it makes, it misses a significant number of positive cases.

GoogleNet and DenseNet-121 showcase higher resilience with F1 scores of 0.83 and 0.85, respectively. Their complex architectures with inception modules and dense connectivity, combined with pretraining, seem to provide an advantage in extracting meaningful patterns even from highly permuted tiles, as evidenced by their substantial accuracy and recall rates.

Also here the Custom CNN, tailored from scratch, achieves perfect precision but has almost null recall (0.01), leading to an extremely low F1 score (0.02). This might indicate that failing to generalize, likely due to its lack of pretraining and perhaps the absence of architectural features that aid in recognizing highly permuted inputs.

Table 5. 20-Tile Permutations Classification Performance Metrics by Architectures

	ResNet-18	GoogleNet	DenseNet-121	Custom CNN
Accuracy	0.54	0.82	0.84	0.52
F1 Score	0.08	0.83	0.85	0.02
Recall	0.04	0.9	0.99	0.01
Precision	0.94	0.76	0.75	1



In summary, ResNet-18, GoogleNet, and DenseNet-121 consistently demonstrated superior classification accuracy and F1 scores, indicating their robustness in handling permuted images. These architectures leverage advanced features such as skip connections, inception modules, and dense connectivity, which enable them to effectively extract and recognize features despite spatial disruptions. Notably, GoogleNet and DenseNet-121 emerged as the top-performing architectures across the permutations, showcasing resilience to spatial disruptions and maintaining high classification accuracy throughout the experiments.

4.1.2 Evaluating Permutation Impact Across Diverse Architectural Designs

Initially, we explored how network characteristics, including depth, complexity, architectural features, and the influence of pretraining, respond to various permutations. Now, we will delve into the impact of

tile resolution, examining how changes in tile size affect model performance.

In the performance of ResNet-18 across different permutations,as shown in Table 6, we observe a trend where increased permutation granularity corresponds to varying impacts on classification metrics. Notably, the network maintains robust performance up to a 2-tile permutations, with minimal decrease in accuracy and F1 score, suggesting that its deep architecture and skip connections effectively manage slight disruptions in spatial information.

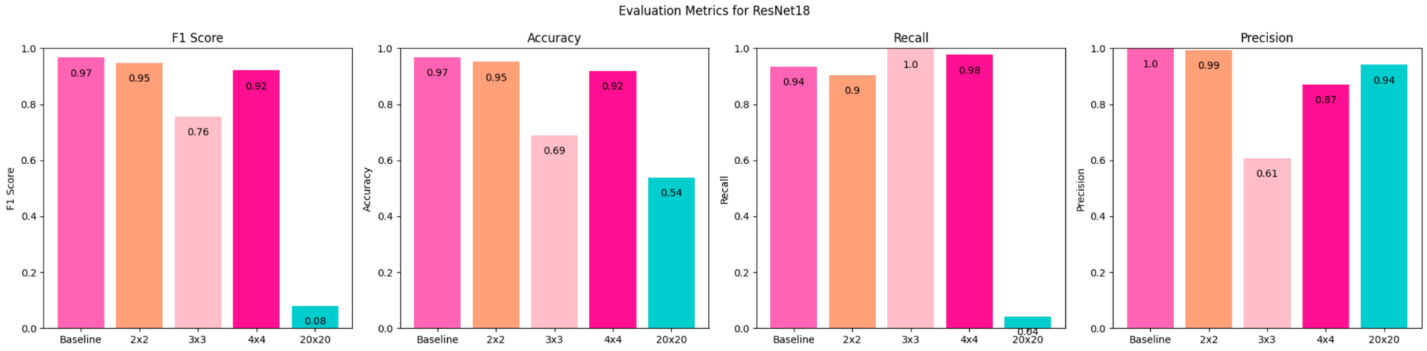
However, at a 3-tile permutations, there's a notable drop in accuracy and F1 score, implying that the network, while deep and complex, starts to struggle with more significant permutations that scatter critical spatial features. This could be due to the resolution of tiling approaching the limits of the network's ability to integrate dispersed features effectively, despite the aid of residual learning in mitigating the vanishing gradient problem.

At 4-tile permutations, the network demonstrates a rebound in performance metrics, potentially indicating an adaptation to this level of permutation. It might be the case that ResNet-18's architecture is capable of learning new spatial relationships at this granularity, aided by the residual connections that preserve information flow.

Yet, at the extreme granularity of 20-tile permutations, the network's performance significantly deteriorates, with a drastic reduction in recall, indicating an inability to detect true positives effectively. This suggests that beyond a certain threshold of permutation complexity, the depth and architectural sophistication of ResNet-18 cannot compensate for the severe disruption of image integrity, leading to a substantial decline in classification capability. Despite its robust features and pretraining, the network is not invulnerable to the challenges posed by extensive permutation.

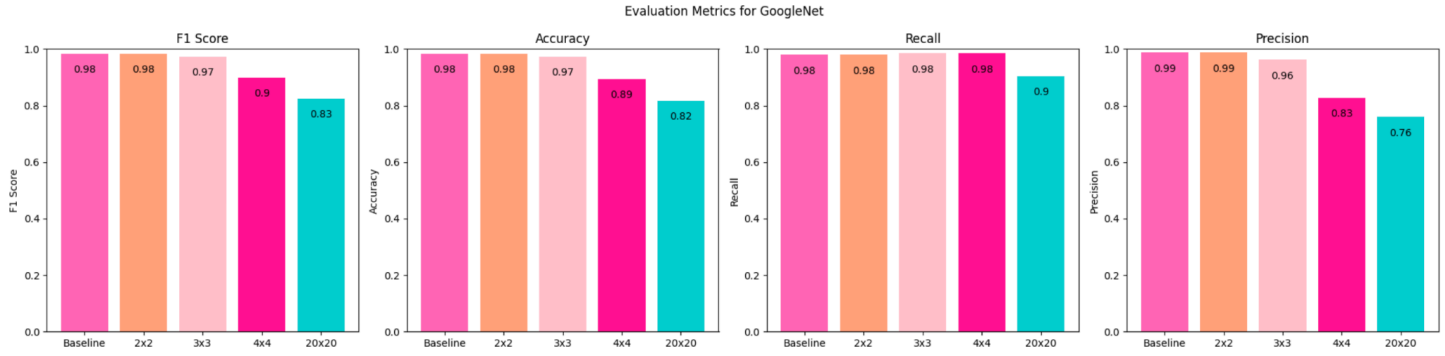
Table 6. ResNet18 Classification Performance Metrics by Permutation

	baseline	2x2	3x3	4x4	20x20
Accuracy	0.97	0.95	0.69	0.92	0.54
F1 Score	0.97	0.95	0.76	0.92	0.08
Recall	0.94	0.9	1	0.98	0.04
Precision	1	0.99	0.61	0.87	0.94



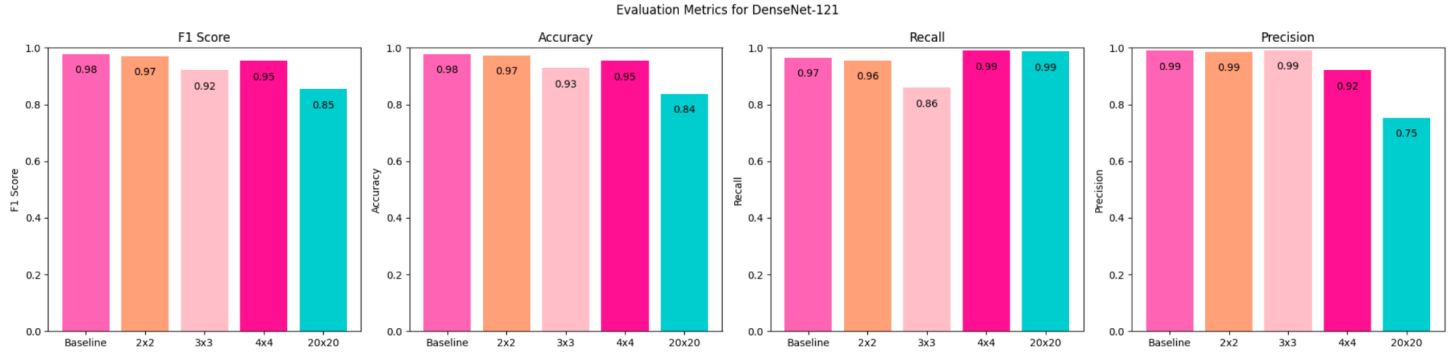
GoogleNet maintains consistently high performance across various permutation sizes, as shown in Table 8, with only a slight decrease as permutation complexity increases. Its architectural depth, combined with the multi-scale feature processing enabled by inception modules, appears to retain spatial information effectively even when image tiles are rearranged. The slight decline in precision from 0.99 to 0.76 for the 20-tile permutations suggests a growing challenge in distinguishing between classes as the permutation complexity increases. The network's ability to adapt to these permutations may benefit from its extensive pretraining, which likely provides a broad feature base that improves its robustness to spatial disruptions.

Table 7. GoogleNet Classification Performance Metrics by Permutation					
	baseline	2x2	3x3	4x4	20x20
Accuracy	0.98	0.98	0.97	0.89	0.82
F1 Score	0.98	0.98	0.97	0.9	0.83
Recall	0.98	0.98	0.98	0.98	0.9
Precision	0.99	0.99	0.96	0.83	0.76



For DenseNet-121, a deeper network with dense connectivity, we observe consistent high performance across different permutations, as shown in Table 9. In the scenario of 4-tile permutations, DenseNet-121 continues to exhibit robust performance even with larger tile sizes. Despite the increased spatial disruptions caused by larger tiles, DenseNet-121 maintains high accuracy and F1 score, demonstrating its resilience to such perturbations. The dense connectivity within the network allows for efficient information flow, facilitating effective feature propagation across layers. This enables DenseNet-121 to handle the challenges posed by larger permuted tiles while still achieving strong classification metrics. The network's ability to adapt to varying levels of spatial disruptions highlights its effectiveness in extracting and utilizing features from permuted images, contributing to its overall robustness in classification tasks.

Table 8. DenseNet-121 Classification Performance Metrics by Permutation					
	baseline	2x2	3x3	4x4	20x20
Accuracy	0.98	0.97	0.93	0.95	0.84
F1 Score	0.98	0.97	0.92	0.95	0.85
Recall	0.97	0.96	0.86	0.99	0.89
Precision	0.99	0.99	0.99	0.92	0.75

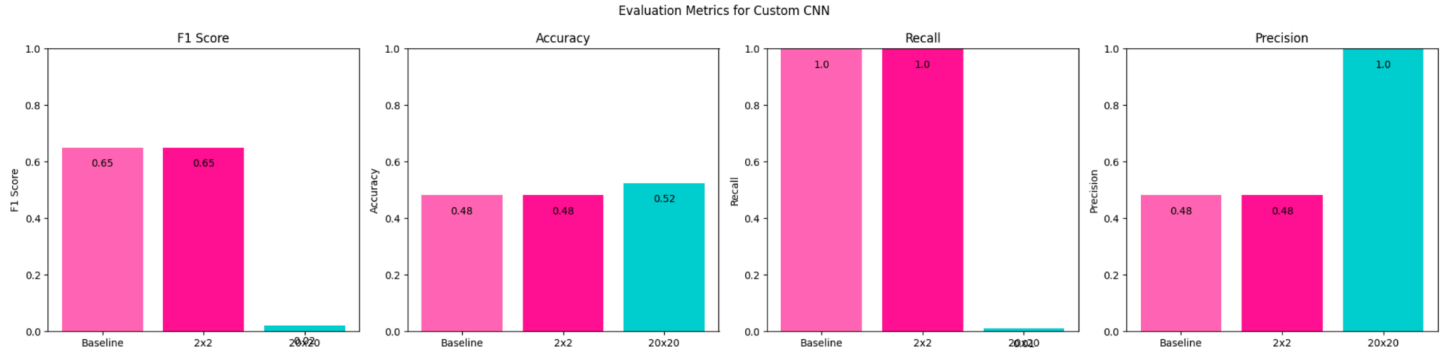


For the Custom CNN architecture, characterized by its sequential structure and trained from scratch, we observe notable differences in performance across different permutations, as shown in Table 10. In the baseline scenario, the Custom CNN achieves moderate accuracy and F1 score, but with a perfect recall and compromised precision, indicating a tendency to classify most instances as positive. When subjected to 2-tile permutations, the network's performance remains consistent, suggesting its limited adaptability to spatial disruptions caused by smaller tile sizes. However, in the case of 20-tile permutations, the Custom CNN's performance deteriorates significantly, with a sharp decline in accuracy, F1 score, and

recall. The network struggles to generalize to highly permuted images, leading to a drastic reduction in classification metrics. The absence of pretraining and advanced architectural features may hinder the Custom CNN's ability to effectively extract meaningful features from permuted tiles, resulting in poor performance in handling spatial disruptions introduced by larger permutations.

Table 9. Custom CNN Classification Performance Metrics by Permutation

	baseline	2x2	20x20
Accuracy	0.48	0.48	0.52
F1 Score	0.65	0.65	0.02
Recall	1	1	0.02
Precision	0.48	0.84	1



In summary, it appears that as the complexity of permutations increases, the task becomes more challenging for the architectures, resulting in decreased performance. This trend was observed particularly in architecture with more parameters, ResNet-18, which exhibited worse results on permutations with higher resolutions. Conversely, deeper and more complex architectures like GoogleNet and DenseNet showed better resilience to spatial disruptions. DenseNet's use of dense connectivity, where each layer is connected to every other layer in a feed-forward manner, contributes to its parameter efficiency. GoogleNet's inception modules facilitate multi-scale feature processing, aiding in the capture of spatial hierarchies efficiently. Pre-trained architectures have been shown to be more effective- the custom CNN model lacks the depth and complexity seen in pre-trained architectures and has completely crashed.

4.1.3 Analysis of False Positive and False Negative Errors

We undertake a detailed investigation into the occurrence of false positive (FP) and false negative (FN) errors within the framework of our vehicle classification task. FP errors occur when instances of bikes are incorrectly classified as cars, while FN errors arise when actual cars are mistakenly classified as bikes. By scrutinizing these errors, we aim to elucidate the specific challenges encountered by our classification models.

ResNet-18: Demonstrates resilience to false positives at most permutation levels but struggles with false negatives, particularly at 20x20.

DenseNet-121: Shows a significant increase in false positives at 20x20, indicating susceptibility to extensive permutation complexities.

GoogleNet: Exhibits relatively consistent false positive rates across permutations but experiences an increase in false negatives at 20x20, suggesting difficulty in accurately identifying cars amidst highly granular permutations.

Custom CNN: Shows high false positives at baseline and 2x2, suggesting limitations in handling spatial disruptions without pretraining or advanced architectural features.

Across all architectures and permutations, there is a noticeable trend of increasing error rates, both FP and FN, as the complexity of permutations rises. This suggests that more intricate permutations disrupt the models' ability to accurately classify images, leading to higher rates of misclassification. The most significant increase in errors is observed at the 20-tile permutations level, indicating that extremely granular permutations pose the greatest challenge to the models.

False Positives (Bikes misclassified as Cars): The occurrence of false positives varies across permutations and architectures. In less complex permutations (baseline and 2x2), false positive rates are relatively low or negligible. However, as the complexity increases, particularly at 3x3, 4x4, and 20x20, false positives become more prevalent. The rise in false positives could be attributed to the fragmentation of spatial features caused by permutations, leading the model to misinterpret bike-related features as indicative of cars. Additionally, the lack of contextual information due to fragmented tiles may contribute to misclassifications.

False Negatives (Cars misclassified as Bikes): Similarly, false negative rates fluctuate across permutations and architectures. While false negatives are relatively low at baseline and 20-tile permutations for some architectures, they spike notably at intermediate permutation complexities (3x3 and 4x4). False negatives may occur when the model fails to recognize car-related features due to spatial disruptions caused by permutations. In more granular permutations, such as 20x20, false negatives are prominent, indicating a significant challenge in accurately identifying cars amidst extensive image permutations.

Overall, this underscores the challenges posed by image permutations on classification performance, with increasingly complex permutations leading to higher error rates. Specific architectures exhibit varying degrees of resilience to permutation-induced disruptions, with deeper and more complex networks generally outperforming simpler counterparts.

Table 10. False Classification Analysis for ResNet18 by tiles

	baseline	2x2	3x3	4x4	20x20
FP - Bikes incorrectly classified as Cars	0%	0.48%	59.76%	13.49%	0.24%
FN - Cars incorrectly classified as Bikes	6.49%	9.61%	0.26%	2.34%	95.84%

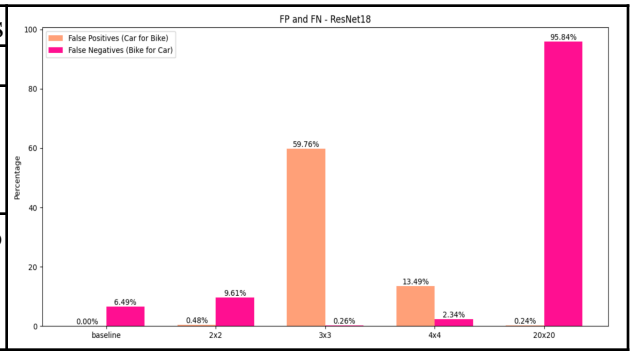


Table 11. False Classification Analysis for Denset-121 by tiles

	baseline	2x2	3x3	4x4	20x20
FP - Bikes incorrectly classified as Cars	0.96%	1.20%	0.72%	7.95%	30.12%
FN - Cars incorrectly classified as Bikes	3.12%	4.42%	14.03%	1.04%	1.3%

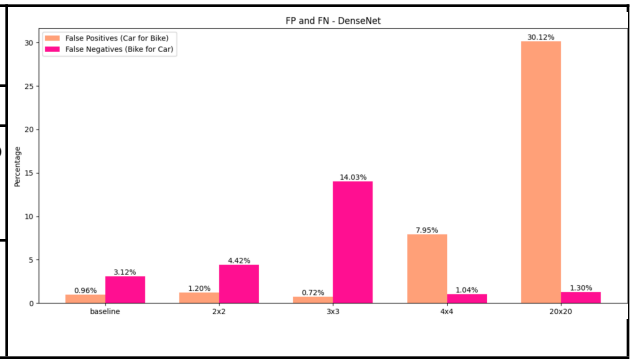


Table 12. False Classification Analysis for GoogleNet by tiles					
	baseline	2x2	3x3	4x4	20x20
FP - Bikes incorrectly classified as Cars	1.20%	1.20%	3.61%	19.04%	26.51%
FN - Cars incorrectly classified as Bikes	2.81%	2.08%	1.56%	1.56%	9.61%

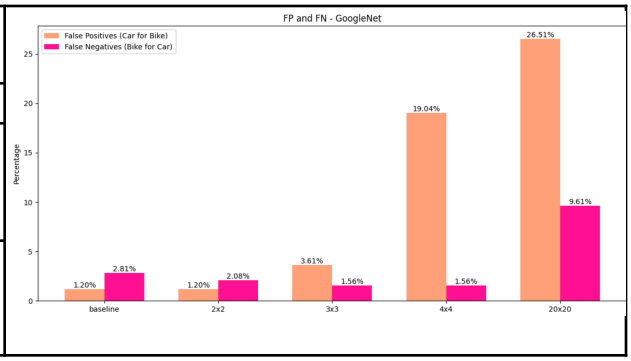
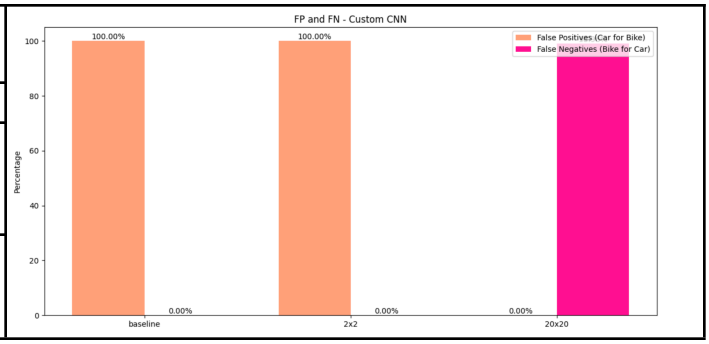


Table 13. False Classification Analysis for Custom CNN by Permutation			
	baseline	2x2	20x20
FP - Bikes incorrectly classified as Cars	100%	100%	0%
FN - Cars incorrectly classified as Bikes	0%	0%	100%



4.2 Enhancing Performance: Exploring Optimization Strategies for DenseNet and GoogleNet

In this section, we will delve deeper into the performance of DenseNet and GoogleNet, two models that have emerged as the top performers across most types of permutations in the previous sections. Our focus will be on exploring various hyperparameters and analyzing their effects on the performance of these models. We aim to gain insights into how different hyperparameters influence the performance of DenseNet and GoogleNet, ultimately contributing to a better understanding of their overall performance characteristics.

4.2.1 Batch Normalization's Impact on Performance

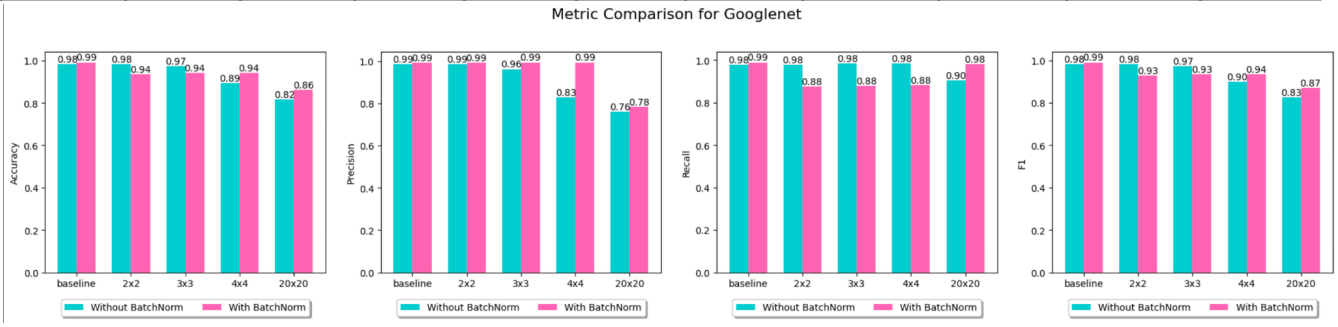
Batch normalization is a vital technique in deep learning, designed to enhance the training efficiency and stability of neural networks. It standardizes the activations of each layer within a mini-batch, mitigating issues like internal covariate shift and promoting faster convergence. In this section, we investigate the impact of batch normalization on the performance of DenseNet and GoogleNet, aiming to uncover how this technique influences their effectiveness across different scenarios.

In GoogleNet For the baseline permutations, batch normalization seems to enhance (not significantly) accuracy and F1 scores, this improvement suggests that in this less complex permutation, batch normalization aids in stabilizing the network's internal distributions, thereby enhancing its predictive performance.

However, in the 2x2, 3x3 and 4-tile permutations, while batch normalization improves precision, it leads to a decrease in both accuracy and F1 score, alongside a noticeable drop in recall. This indicates that while batch normalization helps the network make more precise predictions, it may also lead to a higher rate of false negatives, suggesting a potential overfitting issue or a too aggressive normalization that might be suppressing useful feature variability.

In the case of the highly complex 20-tile permutations, batch normalization significantly boosts the network's performance in all metrics, most notably in recall, which jumps from 0.90 to 0.98. This substantial improvement suggests that in scenarios of extreme permutation complexity, where the risk of overfitting or internal covariate shift is heightened, batch normalization provides a critical stabilizing effect that allows the network to maintain its discriminative power.

Table 14. Performance Comparison of GoogleNet with and without Batch Normalization										
	baseline		2x2		3x3		4x4		20x20	
	Standard	With BN	Standard	With BN	Standard	With BN	Standard	With BN	Standard	With BN
Accuracy	0.98	0.99	0.98	0.94	0.97	0.94	0.89	0.94	0.82	0.86
F1 Score	0.98	0.99	0.98	0.93	0.97	0.93	0.9	0.94	0.83	0.87
Recall	0.98	0.99	0.98	0.88	0.98	0.88	0.98	0.88	0.9	0.98
Precision	0.99	0.99	0.99	0.99	0.96	0.99	0.83	0.99	0.76	0.78



Initially, for the baseline permutation, DenseNet-121 without BN achieves an impressive accuracy and F1 score of 0.98. Upon the application of BN, there is a slight decrease in accuracy to 0.94 and in F1 score to 0.93. This marginal dip suggests that for images with standard spatial coherence, the regularization effect of BN might slightly constrain the model's ability to capitalize on the inherent structure of the data, which is already well-handled by the network's innate design.

As we progress to the 2-tile permutations, the application of BN keeps the accuracy constant at 0.97 and decreases the F1 score marginally from 0.97 to 0.96. This maintenance of performance indicates that BN continues to offer a stabilizing effect without significantly impacting the network's ability to adapt to minor spatial perturbations.

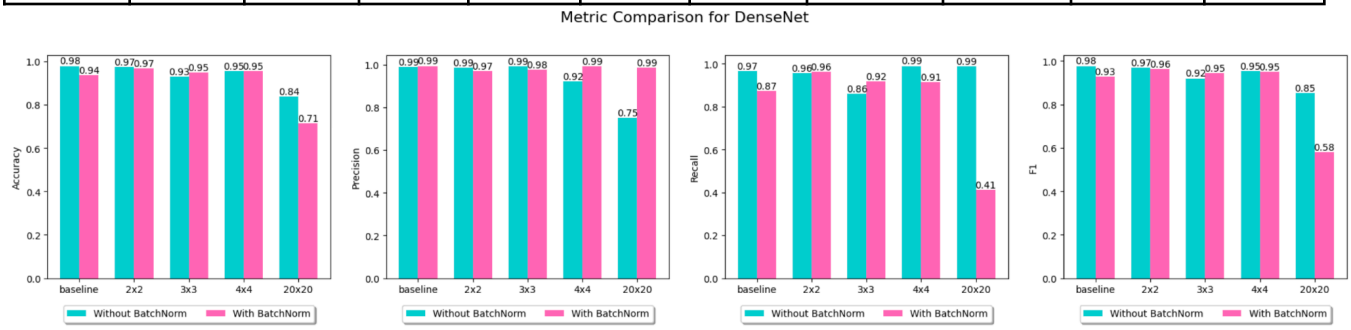
The trend takes an intriguing turn at the 3-tile permutations, where BN appears to bolster the network's robustness, evidenced by a rise in the F1 score from 0.92 to 0.95. This increase suggests that BN is particularly beneficial in this permutation complexity, possibly by aiding the network in better handling the increased variability and ensuring more stable feature activation distributions.

In the 4-tile permutations, BN maintains the F1 score at a high level of 0.95 and sustains accuracy at 0.95, reinforcing the notion that BN is adept at facilitating the network's generalization capability even as the permutations grow more complex.

However, the advantage of BN significantly diminishes at the 20-tile permutations. Here, we observe a considerable drop in accuracy from 0.84 to 0.71 and in the F1 score from 0.85 to 0.58 with BN. Moreover, recall dramatically decreases from 0.89 to 0.41, while precision remains high at 0.99. This steep decline in recall with BN indicates a critical limitation of the technique in the context of highly permuted images, where the standardization of activations may be excessively suppressing the nuanced feature variances necessary for accurate classification, leading to a high number of false negatives.

The hypothesis that emerges from this pattern is that BN's effectiveness is contingent upon the complexity of the image permutations. In cases of moderate permutations (2x2 and 3x3), BN aids in performance consistency and slight improvement. However, at extreme permutation granularities (20x20), the homogenization effect of BN may obliterate the subtle but crucial features necessary for distinguishing classes, thus diminishing the model's discriminative capacity. This suggests a nuanced role of BN in dense architectures like DenseNet-121, where its regularization benefits are overshadowed by the detrimental impact on the network's sensitivity to essential feature distinctions in highly permuted images.

Table 15. Performance Comparison of DenseNet-121 with and without Batch Normalization										
	baseline		2x2		3x3		4x4		20x20	
	Standard	With BN	Standard	With BN	Standard	With BN	Standard	With BN	Standard	With BN
Accuracy	0.98	0.94	0.97	0.97	0.93	0.95	0.95	0.95	0.84	0.71
F1 Score	0.98	0.93	0.97	0.96	0.92	0.95	0.95	0.95	0.85	0.58
Recall	0.97	0.87	0.96	0.96	0.86	0.92	0.99	0.91	0.89	0.41
Precision	0.99	0.99	0.99	0.97	0.99	0.98	0.92	0.99	0.75	0.99



Batch normalization demonstrates a complex relationship with network performance in DenseNet-121 and GoogleNet across varying permutations. For both models, batch normalization generally maintains or improves performance on less complex permutations. In the densest permutation (20x20), batch normalization's impact diverges; it improves GoogleNet's recall significantly, suggesting enhanced generalization in the face of extreme spatial permutations. Conversely, DenseNet-121 sees a reduction in performance with batch normalization, particularly in recall, indicating a possible oversimplification of features essential for classification in highly permuted images. These findings imply that batch normalization's utility is not universal but contingent on the specific architecture and permutation complexity, beneficial in some scenarios while detrimental in others, possibly due to how batch normalization interacts with the model's intrinsic handling of feature variance.

4.2.2 Epoch Exploration

As we continue to enhance the performance of DenseNet-121 and GoogleNet, the number of training epochs emerges as a critical hyperparameter worth investigating. The epoch count, denoting the number of times the learning algorithm will work through the entire training dataset, plays a pivotal role in how well a neural network learns and generalizes from the given input data. Adjusting the number of epochs can lead to significant differences in model performance, potentially mitigating overfitting or underfitting depending on the permutation complexity and network architecture. In this section, we will systematically explore how varying the epoch count influences the robustness and accuracy of DenseNet-121 and GoogleNet across different image permutations, aiming to discern the optimal training duration that harmonizes model proficiency with computational efficiency.
**The data that will be presented later refers to the evaluation measures of the test set.

When analyzing the influence of varying the number of epochs on GoogleNet's performance, we uncover a nuanced trend where the relationship between epoch count and classification accuracy is not linear nor consistent across all types of permutations.

At the outset, a single epoch yields suboptimal results; this is expected as the network is just beginning to learn. Increasing the number of epochs generally leads to an improvement in accuracy and F1 score, up to a point. For the baseline and 2-tile permutations, there's some improvement as the epochs increase, with the highest accuracy and F1 score observed at six epochs, indicating that more extensive training enhances the model's ability to capture and generalize from the data's spatial structure.

However, this trend is not mirrored in more complex permutations. For instance, the 3-tile permutations sees a dip in performance when moving from two to three epochs but then regains performance at six epochs. The 4-tile permutations exhibits a similar fluctuation, with performance peaking at four epochs.

These variations suggest that certain permutation complexities may require specific epoch durations to optimize performance, and overtraining could be detrimental, potentially due to overfitting.

The most pronounced variability is observed in the 20-tile permutations, where the accuracy initially plummets at two epochs, followed by a substantial gain at three epochs, and then a plateauing effect as we advance to six epochs. This indicates that for highly complex permutations, a more delicate balance is required, as the model may rapidly learn the permuted patterns initially but then struggle to further improve due to the complexity of the task.

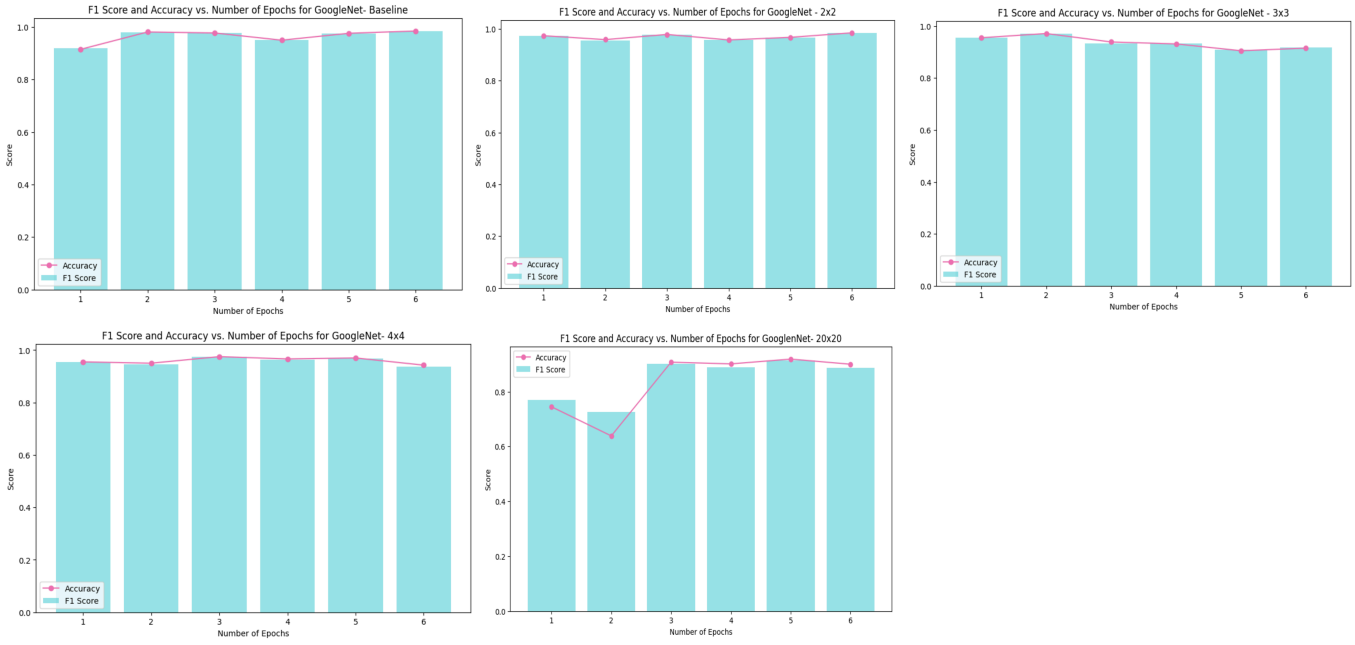
May the initial gains from increased epochs are due to the model effectively learning the reorganized spatial features, but as training progresses, the network might start memorizing the training data rather than learning to generalize, hence the observed performance plateaus. The architecture's depth and complexity allow for the extraction of intricate features; however, the learning process can become saturated, implying that there's an optimal point at which additional epochs do not equate to improved performance. This phenomenon underscores the importance of early stopping in training to prevent overfitting, particularly for complex permutations where the risk is higher.

Table 16. Accuracy (test set) vs. Number of Epochs for GoogleNet

	baseline	2x2	3x3	4x4	20x20
1	0.915	0.974	0.955	0.955	0.745
2	0.981	0.959	0.971	0.950	0.639
3	0.977	0.979	0.939	0.975	0.908
4	0.95	0.958	0.931	0.966	0.901
5	0.976	0.968	0.905	0.970	0.919
6	0.985	0.985	0.915	0.943	0.900

Table 17. F1 Score (test set) vs. Number of Epochs for GoogleNet

	baseline	2x2	3x3	4x4	20x20
1	0.918	0.973	0.955	0.954	0.770
2	0.98	0.955	0.970	0.946	0.726
3	0.976	0.977	0.933	0.974	0.902
4	0.95	0.957	0.933	0.964	0.890
5	0.975	0.966	0.909	0.968	0.913
6	0.984	0.984	0.918	0.937	0.887



The analysis of DenseNet-121's performance in relation to the number of training epochs indicates a general trend where a moderate amount of training enhances the network's ability to learn and generalize effectively, as shown by an overall improvement in accuracy and F1 score with an increasing number of epochs. This improvement is most notable at six epochs for most permutations, suggesting that DenseNet-121 benefits from extended training periods up to a certain point.

However, this upward trend does not uniformly apply to the most complex permutation, 20x20, where we observe a dramatic drop in both accuracy and F1 score at six epochs. This contrast may be attributed to the overfitting of DenseNet-121 when faced with highly permuted images, indicating that the

network's generalization capabilities diminish when excessively exposed to such complex data during training.

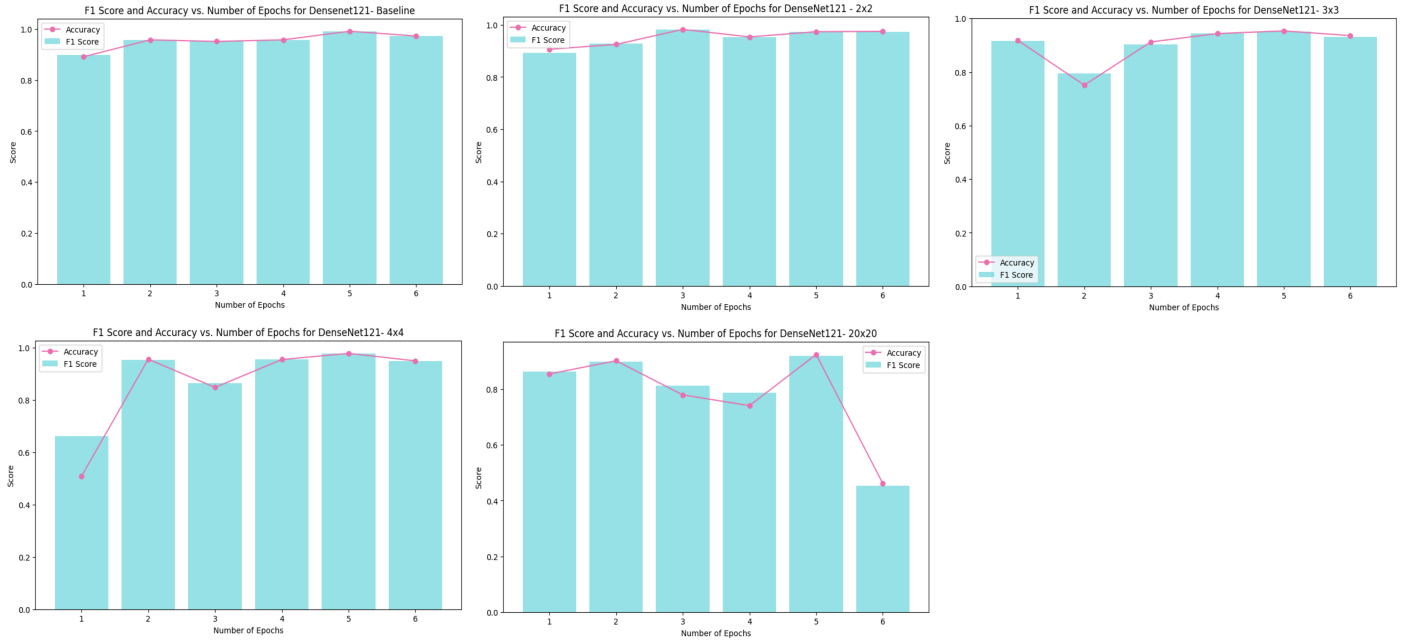
This compatibility between all permutations for DenseNet-121, except for 20x20 at the highest epoch count, can be due to the architecture's dense connectivity, which promotes better feature reuse throughout the network. Such a design could inherently provide a strong foundation for learning detailed patterns without necessitating prolonged training for less complex permutations. However, when permutations are at their most complex, the intricate features learned by the network could potentially lead to a memorization of the training data rather than true generalization, thus resulting in the performance decline seen at higher epochs. This suggests a nuanced relationship between training duration and permutation complexity, underscoring the importance of optimizing.

Table 18. Accuracy vs. Number of Epochs for DenseNet121

	baseline	2x2	3x3	4x4	20x20
1	0.891	0.906	0.919	0.508	0.855
2	0.957	0.925	0.751	0.955	0.903
3	0.951	0.983	0.913	0.848	0.780
4	0.957	0.954	0.944	0.954	0.741
5	0.991	0.974	0.954	0.978	0.925
6	0.9725	0.975	0.936	0.949	0.461

Table 19. F1 Score vs. Number of Epochs for DenseNet121

	baseline	2x2	3x3	4x4	20x20
1	0.897	0.894	0.916	0.662	0.864
2	0.956	0.927	0.795	0.953	0.899
3	0.951	0.982	0.904	0.863	0.813
4	0.957	0.954	0.944	0.954	0.787
5	0.99	0.973	0.952	0.977	0.921
6	0.971	0.974	0.931	0.948	0.454



In summary, the optimal epoch count for training DenseNet-121 and GoogleNet is contingent upon the architectural traits and the permutation's complexity. For more complex permutations, the influence of epoch number is pronounced—higher epochs are necessary to achieve peak accuracy on the training set. Yet, post-peak, these models experience a sharper performance downturn compared to simpler permutations, indicating a swifter onset of overfitting. This could be due to the networks' detailed feature extraction capacity, which, when overtrained, starts to memorize noise instead of generalizing from the underlying patterns in highly complex permuted images.

**Due to computational limitations, we will restrict the analysis to using three epochs, despite acknowledging that higher epoch counts may yield better performance.

4.2.3 Batch Size Exploration

The exploration of batch size presents another avenue to fine-tune the performance of DenseNet and GoogleNet. Batch size dictates the number of training samples to be processed before the model's internal parameters are

updated. This hyperparameter is influential in determining the gradient's variance and the stability of the learning process. Smaller batch sizes often provide a regularizing effect and better generalization but may increase training time and noise in the gradient estimation. Conversely, larger batch sizes facilitate computational efficiency through parallel processing but can sometimes lead to poorer convergence of the optimization algorithm. We will be examining batch sizes ranging from 4 to 64 to capture the broad effects of this hyperparameter on the model's training and validation outcomes.

Upon examining the influence of batch size on GoogleNet's test set performance, it is evident that different permutations respond uniquely to changes in batch size. For the baseline and 4-tile permutations, the highest accuracy and F1 scores are achieved with a batch size of 16. This could suggest that a mid-range batch size provides a balance between the regularization benefits of smaller batches and the stability and efficiency of larger batches, particularly for these permutation complexities.

Contrastingly, for 2x2, 3x3, and 20-tile permutations, the optimal batch size also appears to be 16, which consistently provides superior performance across these varied permutation complexities. The trend suggests that GoogleNet's architecture, which is capable of capturing multi-scale information, might benefit from a batch size that offers enough data to effectively estimate gradients without introducing significant noise.

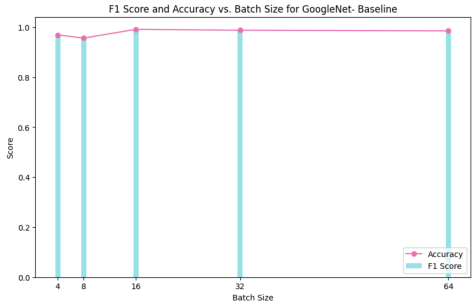
The consistency of batch size 16 in yielding the best results across all permutations highlights the importance of a balanced batch size that neither overwhelms the system with information nor starves it. It implies that for the GoogleNet architecture, batch size 16 strikes the right equilibrium for learning efficiently and generalizing well from the training data, irrespective of permutation complexity. This indicates that while smaller batches might be beneficial for networks learning simpler patterns, a moderately large batch size is more effective for GoogleNet across a spectrum of complexities, possibly due to its particular learning dynamics and how they interact with the batch-induced variance in the optimization process.

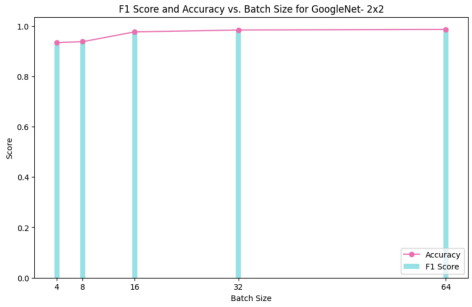
Table 20. Accuracy (test set) vs. Batch Size for GoogleNet

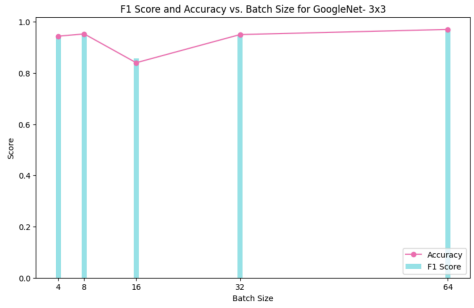
	baseline	2x2	3x3	4x4	20x20
4	0.969	0.934	0.944	0.905	0.774
8	0.956	0.938	0.953	0.964	0.780
16	0.991	0.976	0.840	0.970	0.866
32	0.988	0.984	0.950	0.895	0.875
64	0.985	0.986	0.970	0.953	0.891

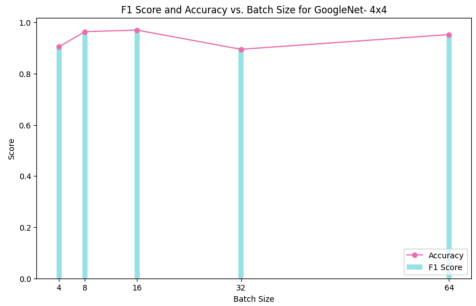
Table 21. F1 Score (test set) vs. Batch Size for GoogleNet

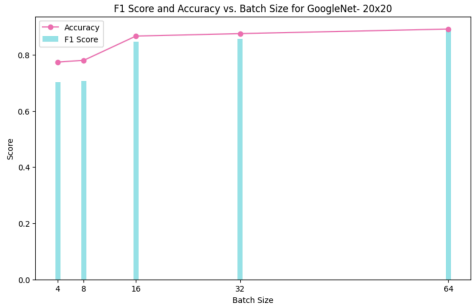
	baseline	2x2	3x3	4x4	20x20
4	0.968	0.928	0.944	0.909	0.703
8	0.956	0.931	0.952	0.962	0.706
16	0.991	0.975	0.857	0.969	0.846
32	0.987	0.983	0.949	0.901	0.856
64	0.984	0.986	0.968	0.949	0.88











DenseNet-121's performance across different batch sizes reveals a discernible trend that aligns with the permutation complexity. For baseline and 4-tile permutations, the largest batch size of 64 leads to the highest accuracy and F1 scores. This suggests that DenseNet-121's architecture, known for its dense connectivity, may benefit from the stability and less noisy gradient estimates provided by larger batch sizes when the data has either uncomplicated patterns or moderately complex structures that still preserve some spatial relationships.

Conversely, for 2x2, 3x3, and particularly the highly permuted 20x20 images, a batch size of 32 appears to be the most effective, suggesting that a balance between stability and the beneficial noise of smaller batches is crucial for learning the disrupted spatial features inherent in these permutations. The smaller batch size may introduce a degree of stochasticity that aids the network in navigating the complex landscape of the loss function, allowing for better generalization when facing more disruptive permutations.

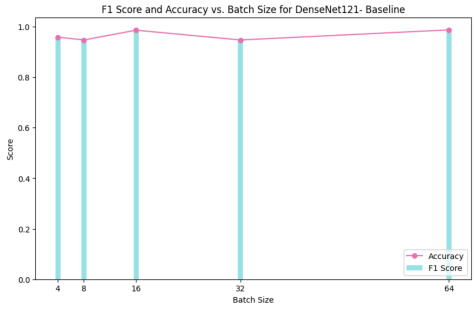
Interestingly, the sharp performance decline seen with the largest batch size in the 20-tile permutations underscores the challenges DenseNet-121 faces with excessive permutation complexity. It seems that too large a batch size may smoothen the loss landscape to the point where the subtleties of highly permuted features are glossed over, leading to poorer generalization capabilities.

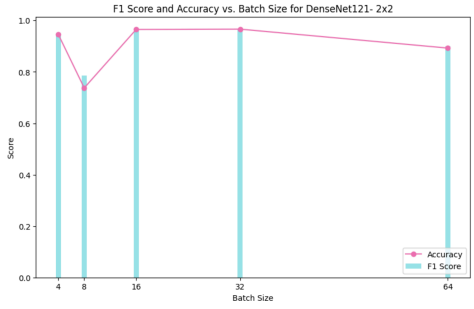
Table 22. Accuracy (test set) vs. Batch Size for DenseNet-121

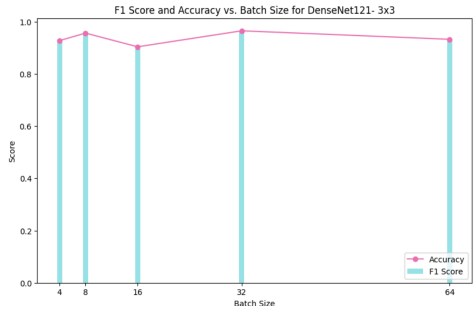
	baseline	2x2	3x3	4x4	20x20
4	0.959	0.946	0.928	0.720	0.798
8	0.948	0.738	0.956	0.930	0.926
16	0.986	0.965	0.904	0.855	0.954
32	0.948	0.966	0.965	0.790	0.964
64	0.988	0.893	0.933	0.969	0.879

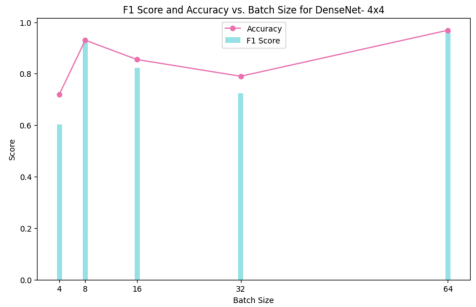
Table 23. F1 Score (test set) vs. Batch Size for DenseNet-121

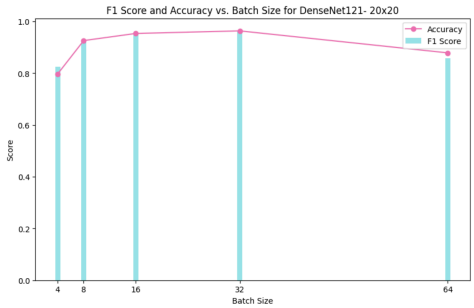
	baseline	2x2	3x3	4x4	20x20
4	0.958	0.946	0.928	0.603	0.826
8	0.943	0.785	0.955	0.932	0.929
16	0.986	0.964	0.893	0.824	0.952
32	0.948	0.966	0.964	0.723	0.961
64	0.987	0.900	0.934	0.968	0.857











In summary, the exploration of batch size effects on DenseNet-121 and GoogleNet revealed nuanced differences in their performance across various permutation complexities. GoogleNet consistently

performed best with a moderate batch size of 16, benefiting from a balance between gradient stability and computational efficiency. In contrast, DenseNet-121 showed a preference for larger batch sizes (64) with simpler patterns but smaller batch sizes (32) for more complex spatial features, reflecting its reliance on a delicate balance between stability and beneficial noise. These differences likely stem from the architectural variances between the two models. The sharp decrease in performance observed with the 20-tile permutations suggests that excessive complexity can overwhelm the models, particularly when utilizing larger batch sizes, hindering their ability to effectively generalize.

4.2.4 Learning Rate Exploration

In this section, we're going to check out how different learning rates—like 0.1, 0.01, 0.001, and 0.0001—affect how well our models train and perform. Learning rate is like the speed at which our models learn from the data. We'll try out different rates to see which ones help our DenseNet and GoogleNet models learn best, making sure they learn well without going too fast or too slow.

When we examine the learning rates for GoogleNet, as shown in Tables 24 and 25, we notice that lower rates lead to higher accuracy and F1 scores across various permutations. Specifically, the rate of 0.0001 consistently outperforms the higher rates. This pattern suggests that a slower learning pace allows the model to adjust gradually without overshooting optimal weights during training, which is crucial for handling the intricate variations caused by image permutations.

Moreover, as the size of the tiles increases, the improvement becomes more significant with lower learning rates. This phenomenon can be attributed to the intricate and nuanced patterns present in permuted tiles. When dealing with larger permutations, such as the 20x20 case, the fine-tuning afforded by a lower learning rate seems particularly beneficial, allowing the model sufficient iterations to integrate the complex, disrupted spatial information without being hampered by the volatility of larger rate adjustments.

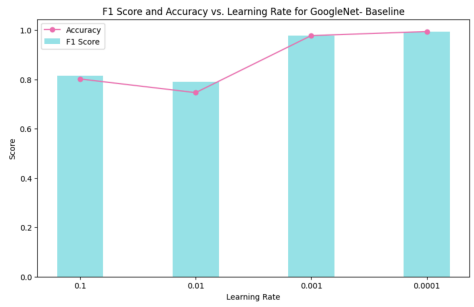
In summary, a lower learning rate of 0.0001 emerges as optimal for GoogleNet across all permutations. This rate enables the model to learn steadily and adapt to the complexities of permutations without sacrificing stability or accuracy. This is especially critical for more complex permutations, where there is a higher risk of overfitting or underfitting, necessitating a delicate balance to achieve optimal generalization.

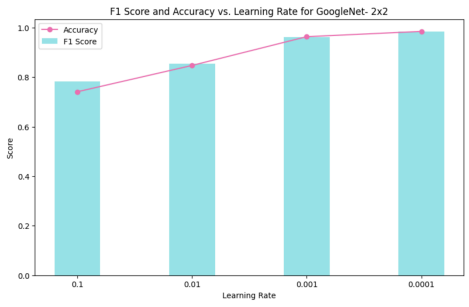
Table 24. Accuracy (test set) vs. Learning Rate for GoogleNet

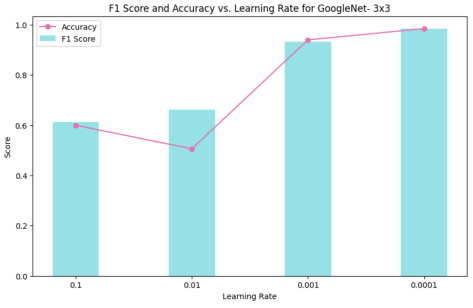
	baseline	2x2	3x3	4x4	20x20
0.1	0.802	0.741	0.600	0.485	0.481
0.01	0.747	0.848	0.506	0.582	0.550
0.001	0.978	0.964	0.940	0.948	0.768
0.0001	0.995	0.985	0.985	0.983	0.900

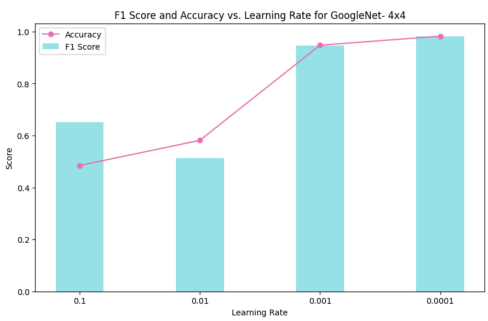
Table 25. F1 Score (test set) vs. Learning Rate for GoogleNet

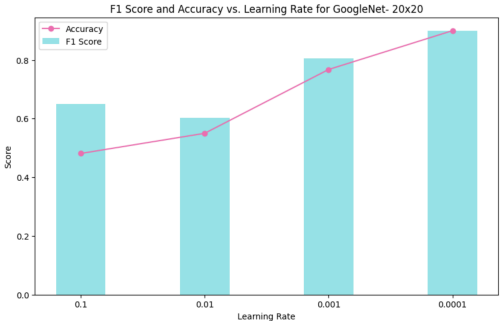
	baseline	2x2	3x3	4x4	20x20
0.1	0.814	0.784	0.613	0.651	0.650
0.01	0.791	0.856	0.661	0.512	0.603
0.001	0.977	0.962	0.934	0.946	0.805
0.0001	0.994	0.984	0.984	0.982	0.900











For DenseNet-121, the comparison between different learning rates and their impact on model performance, as depicted in Tables 26 and 27, does not demonstrate a clear trend in which lower learning rates outperform higher ones across all types of permutations. Notably, the model performs relatively consistently well with the learning rates for the more simpler permutations (baseline through 4x4),. This consistency implies that DenseNet-121 is less sensitive to learning rate variations, possibly due to its efficient use of parameters and dense connectivity, which provide a robust feature extraction and generalization capability.

In the 20x20 permutations, there appears to be a degree of sensitivity to the learning rate, although it is not particularly significant.

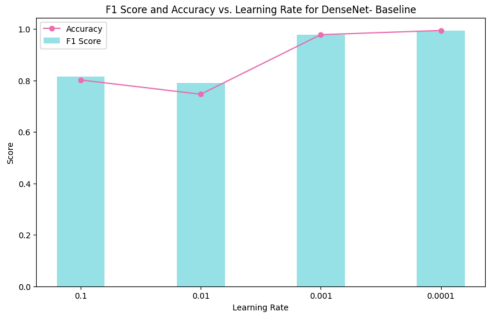
In conclusion, while DenseNet-121 exhibits robustness to learning rate changes across permutations, a moderate learning rate of 0.001 appears to offer the best performance, particularly for handling complex permutations without compromising on the model's accuracy and generalization capabilities.

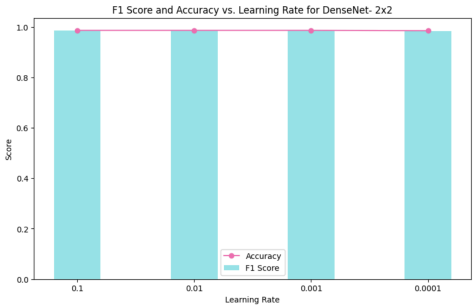
Table 26. Accuracy (test set) vs. Batch Size for DenseNet-121

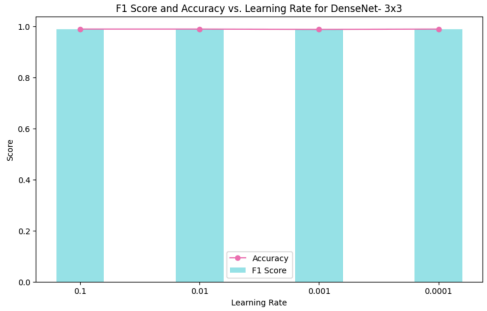
	baseline	2x2	3x3	4x4	20x20
0.1	0.9957	0.986	0.989	0.982	0.896
0.01	0.9946	0.986	0.989	0.983	0.896
0.001	0.9957	0.986	0.988	0.982	0.908
0.0001	0.9957	0.985	0.989	0.983	0.889

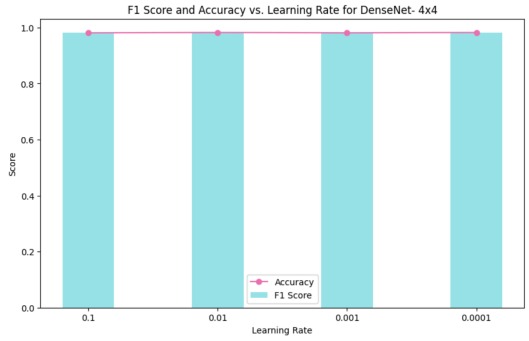
Table 27. F1 Score (test set) vs. Batch Size for DenseNet-121

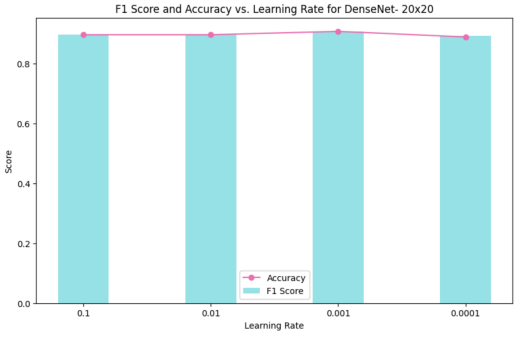
	baseline	2x2	3x3	4x4	20x20
0.1	0.9955	0.986	0.988	0.981	0.897
0.01	0.9943	0.986	0.988	0.982	0.897
0.001	0.9955	0.986	0.987	0.981	0.908
0.0001	0.9955	0.984	0.988	0.982	0.893











In exploring the effects of different learning rates, it's clear that a slower learning rate, particularly 0.0001, greatly benefits GoogleNet across all permutations, enhancing its learning process and ability to handle permuted images. DenseNet-121, however, shows robustness to a range of learning rates, with 0.001 slightly edging out as the most

effective, especially in more complex permutation scenarios. The investigation underscores the importance of carefully tuning learning rates according to the specific architecture and the complexity of the task to optimize neural network performance.

4.2.5 Augmentation Strategies

Building upon our project's objective to assess neural network resilience to image permutations, we now turn to augmentation strategies as a means to bolster classification robustness. As established in the literature review, particularly referencing the study on "MULTI PRETEXT TASK SELF-SUPERVISED LEARNING FOR CULTURAL HERITAGE CLASSIFICATION," such strategies can significantly enhance model performance through exposure to a wider variety of spatial configurations. This next phase will investigate how different augmentation techniques, such as rotation, scaling, and flipping, can aid DenseNet-121 and GoogleNet in better generalizing from permuted images. By systematically applying these techniques, we aim to determine their efficacy in mitigating the adverse effects of permutation, thereby potentially expanding the networks' predictive accuracy and adaptability in the face of altered spatial relationships. We employed random horizontal and vertical flips, random rotations up to 30 degrees, and random adjustments to brightness, contrast, saturation, and hue. These augmentations were applied to both bike and car images, resulting in a total of 600 augmented images (300 for bikes and 300 for cars) added to the dataset and added equal to the train and test sets (so it not change the existing in each). These strategies aim to diversify the dataset and enhance the models' robustness to variations in spatial configurations.

As presented in Table 28 it looks like there is a positive impact of augmentation techniques on the classification performance of GoogleNet. When we examine the original dataset, the model's accuracy and F1 score are impressive across different permutations, with a noticeable drop in the complex 20x20 permutations scenario. The application of augmentation strategies—rotate, scale, and flip—appears to mitigate this decrease for the permuted images, indicating that these techniques enable the model to handle high permutation complexities more efficiently.

For the baseline and 4x4 permutations, augmentation preserves the model's performance. However, there is an improvement when using magnification techniques for other permutations. This improvement is particularly evident in the 2x2 permutation, where both accuracy and F1 score compare favorably to the baseline after augmentation, illustrating that augmentation helps the model generalize better even when spatial features are mildly perturbed. The trend is consistent for 3x3 and 4x4 conversions, albeit with marginal improvements. These subtle increases confirm that magnification helps the network recognize and adapt to changing spatial relationships, improving its resilience to permutation-induced variability.

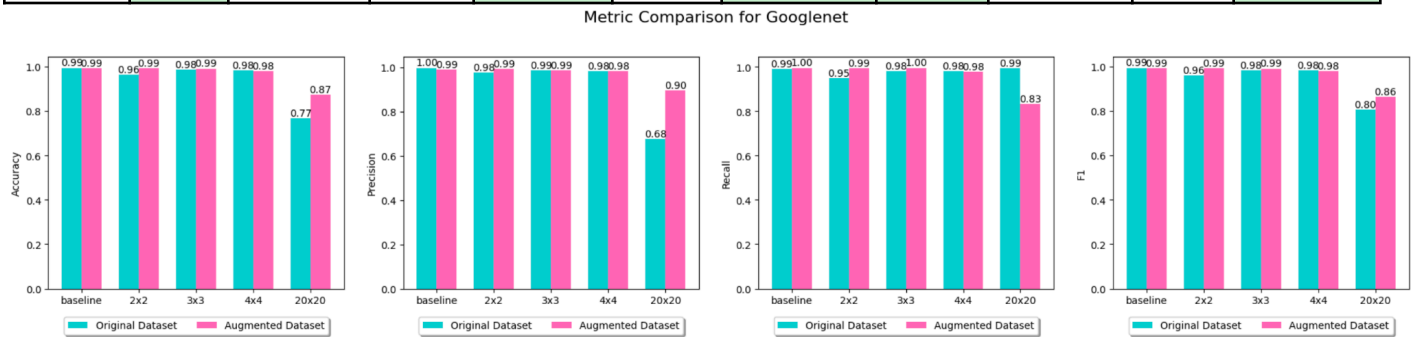
Augmentation strategies have the most significant positive effect on the thinnest 20x20 tiles. Here, accuracy jumps from 0.768 to 0.873, and the F1 score from 0.805 to 0.863 after augmentation. This dramatic improvement suggests that when the spatial structure of images is widely perturbed, exposing the model to a diverse set of spatial configurations during training is essential for effective generalization learning. augmentation appears to act analogously to the increased diversity of the dataset, equipping the model with the ability to navigate more complex permutations successfully.

In conclusion, augmentation strategies prove to be useful and essential for permutations, especially as the complexity of image permutations increases. The consistency in performance improvement across

different tiles after augmentation suggests that these techniques are robust across various levels of permutation complexity.

Table 28. Performance Comparison of GoogleNet with Original and Augmented Dataset

	baseline		2x2		3x3		4x4		20x20	
	Original	Augmented	Original	Augmented	Original	Augmented	Original	Augmented	Original	Augmented
Accuracy	0.995	0.994	0.964	0.994	0.985	0.991	0.983	0.980	0.768	0.873
F1 Score	0.994	0.993	0.962	0.993	0.984	0.991	0.982	0.980	0.805	0.863



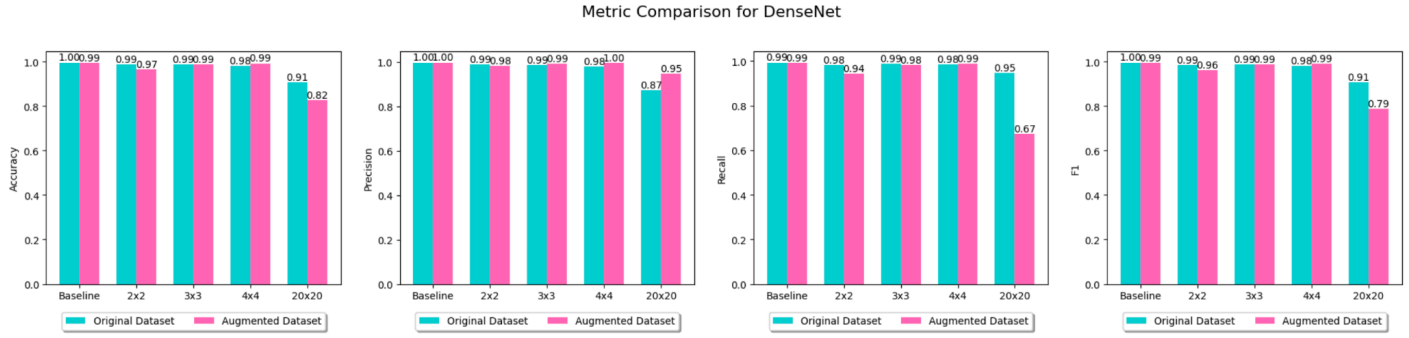
For DenseNet it seems that augmented data does not uniformly improve the performance of the model in all types of permutations. Specifically, for the baseline, 3x3, 2x2 and 20x20 permutations, accuracy and F1 scores are slightly reduced with the augmented dataset, albeit the difference is marginal. This implies that DenseNet's original configurations cope well with these levels of permutation complexity, and augmentation doesn't provide a significant benefit, potentially due to the model's inherent strength in feature extraction and generalization.

However, a notable decline in both accuracy and F1 score is observed with augmentation in the 2x2 and 20x20 permutations. The decrease in performance for the 2x2 permutation might be attributed to the augmentation introducing a level of variability that is not effectively addressed by the model's existing capabilities, possibly adding unnecessary complexity to the learning process. For the 20x20 permutation, the dramatic reduction when using augmented data suggests that DenseNet's performance is substantially impacted by the introduction of additional variance on top of the already complex permutation scenario, which could exceed the model's capacity for effective generalization, leading to overfitting or distraction from the primary task.

In summary, while augmentation is generally considered beneficial for improving neural network robustness, its efficacy may depend on the specific circumstances of the permutation complexity and the existing capabilities of the model. In the case of DenseNet, augmentation strategies need to be judiciously applied, keeping in mind that they might not always lead to improved performance and can even have adverse effects, especially when dealing with highly complex permutations.

Table 29. Performance Comparison of DenseNet with Original and Augmented Dataset

	baseline		2x2		3x3		4x4		20x20	
	Original	Augmented	Original	Augmented	Original	Augmented	Original	Augmented	Original	Augmented
Accuracy	0.996	0.995	0.986	0.965	0.989	0.988	0.983	0.992	0.908	0.825
F1 Score	0.996	0.994	0.986	0.963	0.988	0.988	0.982	0.992	0.908	0.787



In summary, augmentation techniques notably enhance GoogleNet's classification performance in handling complex permutations, such as the 20x20 tiles, reinforcing the model's adaptability to varied spatial configurations. However, DenseNet-121 demonstrates a slight decline with augmented data for certain permutations, suggesting an optimal balance of complexity and generalization already exists within its architecture. These findings highlight that augmentation is not a one-size-fits-all solution, and its effectiveness is highly dependent on the architecture and the complexity of the data it is applied to.

4.3 Feature and Saliency Maps

In this section, we'll examine the feature and saliency maps at various tile resolutions to dissect how the convolutional neural networks process these permutations. We're comparing models pretrained to those further honed on our specific data. Our goal is to understand how tile resolution impacts the network's interpretability and learning.

*Due to the size of each plot and the similarity in conclusions across all resolutions, this section will only display plots for baseline, 3x3, and 20x20 resolutions. Plots for other resolutions are available in the code notebook.

4.3.1 Feature Maps

We anticipated that the GoogleNet pre-trained model would manifest broad and less specialized feature activations, not finely attuned to the specifics of the transformed tile datasets. However, in practice, it appears that the layers remain consistent across all permutations, with notable distinctions observed primarily in the deeper layers. This observation underscores that lower layers predominantly capture simpler and more general features, such as edges and colors, while higher layers, which develop abstract representations, exhibit more significant transformations when the network is trained on specific data. Moreover, the tile size exerts impact on the feature maps- Baseline tiles tend to elicit more uniform and less varied feature activation, representing standard, unaltered images. In contrast, 3x3 tiles, and particularly 20x20 tiles, introduce greater complexity due to permutation, resulting in feature maps characterized by enhanced diversity and specificity. Notably, in 20x20 tiles, even in the early layers, it becomes challenging to discern identifiable shapes, unlike the clearer shapes observed in 3x3 and baseline tiles.