

Deep Learning for Cancer Detection: Optimization, Ensembling, and Data Augmentation

Matthew Muscedere
School of Computer Science
University of Windsor
Windsor, Canada

Salman Youssef
School of Computer Science
University of Windsor
Windsor, Canada

Keigo Katanaga
School of Computer Science
University of Windsor
Windsor, Canada

Azwad Uddin
School of Computer Science
University of Windsor
Windsor, Canada

Abstract—Cancer detection is an active problem in deep learning research. As there have been significant strides in computer vision, techniques began being used to classify images to determine, for example, if a patient is likely to be diagnosed with cancer based on medical imaging. This medical imaging can come in different forms. Of these are histopathological images - microscopic images of tissues from body parts like the lungs or colon. Using a dataset from Kaggle’s Lung and Colon Cancer Histopathological Images competition, we explore the use of Convolutional Neural Networks (CNNs) for lung cancer detection and classification (based on three types of lung cancer) and examine how the performance of these CNNs can be increased through the use of different optimization strategies. The strategies we explored are: hyperparameter optimization, ensembling, and data augmentation. The results saw an increase in performance when implementing some of these techniques, reaching a test loss of less than 0.2% and corresponding accuracy of over 99.8%.

Index Terms—Machine Learning, Deep Learning, Convolutional Neural Networks, Hyperparameter Optimization, Ensembling, Data Augmentation

I. INTRODUCTION

Despite improvements in healthcare, lung cancer is one of the leading causes of cancer-related deaths globally. Improving patient outcomes significantly relies on achieving a proper diagnosis as early as possible. Recently, the development of deep learning tools such as Convolutional Neural Networks (CNNs) has dramatically changed the medical imaging field (Kadam et al., 2021). The ability of CNNs

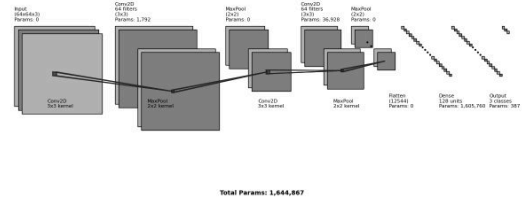


Fig. 1: Base CNN Architecture

to extract complex features from complicated image data makes them ideal for classifying histopathological images; these are intricate microscopic images of tissue that allow pathologists to differentiate between benign and malignant tumors.

This paper describes our work on the task of classification of lung cancer using CNNs. Using a comprehensive dataset from Kaggle’s Lung and Colon Cancer Histopathological Images competition, we focus on the classification of three types of lung tissues: benign, adenocarcinoma, and squamous cell carcinoma. In order to achieve the highest possible accuracy with classification using any of our CNN models, we implement various optimization techniques including hyperparameter optimization, data augmentation, and ensembling techniques.

To achieve this aim, we set up a baseline CNN architecture which we will use in our experiments

(Figure 1).

The paper is organized as follows: Section I reviews relevant literature on CNN applications in medical imaging; Section II describes the tools and technologies utilized; Section III is on the methodology, including the experimental design and optimization techniques; Section IV presents and discusses the results; and Section V provides a conclusion, with references after.

II. LITERATURE REVIEW

Recent advancements in convolutional neural networks (CNNs) have significantly improved their performance in medical image classification, particularly in histopathological imaging. CNNs have demonstrated considerable success in classifying various medical images, including CT scans, MRI, and histopathological slides. Specifically, in the context of lung cancer, CNNs achieved high accuracy when classifying CT images [1]. Additionally, another study on colon cancer indicated that CNNs could effectively differentiate among various disease stages [2]. Given the high stakes in medical diagnostics, where accuracy directly impacts patient outcomes, CNNs have emerged as a robust and reliable tool.

A. Hyperparameter Optimization

Hyperparameter optimization plays a crucial role in fine-tuning CNNs to reach peak performance in cancer detection tasks. Common optimization strategies include Grid Search, Random Search, and Bayesian Optimization. Grid Search was successfully employed in a lung cancer detection study to achieve high accuracy in classifying CT images [3]. Alternatively, Random Search, utilized in breast cancer detection, demonstrated superior performance and efficiency compared to Grid Search [4]. Bayesian Optimization, recognized for its efficiency and adaptive parameter selection, was effectively applied to brain tumor classification from MRI images, highlighting its potential for optimizing CNN performance in medical imaging [5].

B. Data Augmentation

Data augmentation is widely adopted to enhance CNN generalization by artificially increasing the size and diversity of the training dataset. CutMix

a method that creates synthetic training samples by combining image patches from different sources, significantly improved prostate cancer detection accuracy by reducing overfitting [6]. Another effective augmentation method, MixUp, involves blending two images along with their labels, smoothing the decision boundaries to enhance robustness. Research on breast cancer classification found that MixUp improved CNN performance on previously unseen data [7].

C. Ensemble Methods

Ensemble methods combine predictions from multiple CNN models to leverage their complementary strengths, resulting in improved classification accuracy and reliability in cancer detection tasks. Studies across various cancer types including lung and breast cancers have consistently demonstrated the superior performance of ensemble approaches compared to individual CNN models. For example, an ensemble CNN model has shown increased accuracy in predicting lung nodule malignancy [8]. Similarly, a breast cancer classification study utilized meta-learning ensembles to achieve notably higher performance than single models [9]. Moreover, an ensemble combining three architectures (VGG16, ResNet50, InceptionV3) for skin cancer detection highlighted the benefits of leveraging diverse model architectures to enhance reliability and generalizability [10].

III. TOOLS AND TECHNOLOGIES

The development and research methodology for this project involved an excellent combination of contemporary tools and technologies. The experiments were performed in an interactive environment known as Jupyter Notebook, which was particularly useful because it allowed for a cohesive and flexible approach to coding, testing, and visualizing experiment results. The programming language that structured the experimentation and research was Python, which powers all of the powerful libraries and frameworks available to help accelerate the research method. The authors employed the TensorFlow/Keras applications to build and train the convolutional neural network (CNN) models, which provided high-level abstractions, as well as efficient implementations of deep learning. The authors also

used Matplotlib to create effective visualizations that significantly aided the analysis of training metrics and performance trends in the experiments conducted. The authors used NumPy and Pandas for numerical computations and data manipulation, which helped propel the data processing method efficiently and quickly. Also, using GitHub for version control and collaborative code, this suite of technologies provided a great and comprehensive ecosystem for the research endeavor and development of the final paper.

IV. METHODOLOGY

The "baseline" model that was chosen as a starting point for analyzing the results of other techniques looks like this:

It contains two convolutional layers (each containing 64 filters), and each is followed by a 2x2 maxpool layer. Then the data is flattened and fed into a dense layer with 128 units (1.6 million parameters), and then an output (softmax) layer for classification of the three lung tissues, lung-aca, lung-n, and lung-scc representing the three output classes, or lung tissues. lung-aca is adenocarcinoma, lung-n is benign tissue, and lung-scc is squamous cell carcinoma.

The base model used *Adam* for learning rate optimization, and so did all the other models tested in the *Hyperparameter Optimization* section. This ensured a fair playing ground for the models being tested since they all use the same learning rate optimizer, Adam, which is widely considered among the best in today's state-of-the-art.

The model was modified several times throughout this project to test:

- hyperparameter optimization techniques
- ensembling strategies
- data augmentation

Each of these will be analyzed in more detail, and the results from each will be compared using a graph.

A. Hyperparameter Optimization

The project commenced with preprocessing histopathological images collected from a Kaggle dataset consisting of high-resolution images of lung and colon tissues. Resized to 64x64 pixels with three color channels, we struck a balance to

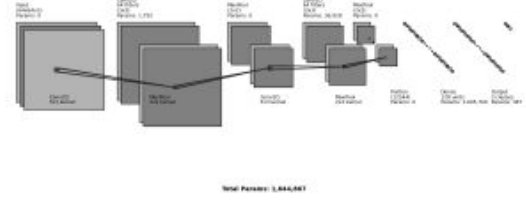


Fig. 2: Base CNN Architecture

optimize computational demands while preserving sufficient detail for classification. The dataset was divided into three random training, validation, and test sets. The training set was 80% of the dataset used for fitting the convolutional neural network (CNN) model. The validation set was 10% of the dataset taken from the training set to track performance while avoiding overfitting the model during development. Finally, the test set was comprised of 20% of the dataset to evaluate the final model after tuning. We put the labels into a one-hot encoded format, as we would be a multi-class classification with a softmax output layer. A flexible CNN model was created using TensorFlow's Keras API. The model structure features two convolutional layers that are each accompanied by a max pooling layer to extract features from the images, which includes both simple and complex features. The first convolutional layer extracts low-level features, while the second layer, which has double the number of filters, observes more complex, abstract features. The resulting output from the convolutional layers is flattened and serve as input to a dense layer to learn complex combined features. The dropout layer is implemented to reduce overfitting by randomly deactivating some percentage of neurons during training. The model culminates with an output layer containing a softmax activation that assigns class probabilities to each of the three classes of lung tissue: benign (benign hyperplasia only with no evidence of malignancy), adenocarcinoma and squamous cell carcinoma. The model was compiled with the Adam optimizer and multiple hyperparameters (including the number of filters, kernel size, number of dense units, dropout, and learning rate) were designated as hyperparameters that would be variable for experimentation. To identify the optimal

hyperparameters, three distinct optimization strategies were employed: Grid Search, Random Search, and Bayesian Optimization.

- **Grid Search** - an exhaustive search over a predefined grid of hyperparameter values. This method guarantees to evaluate all combinations but is extremely time-consuming as dimensionality grows. Grid search often wastes effort on less important parameters; empirically, many hyperparameters are not equally sensitive, making full grids inefficient (Bergstra Bengio 2012). Its main benefit is thoroughness, but the combinatorial explosion in CNNs (which have many tunable parameters) makes it impractical except for very limited ranges.
- **Random Search** - This method specifies a range of values for each hyperparameter and randomly selects combinations from those ranges. Rather than exhaustively checking all possible options, it samples a fixed number of random combinations, which often leads to good results faster than grid search, especially when only a few hyperparameters significantly affect performance.
- **Bayesian Optimization** - This technique uses probability and statistics to model the performance of the objective function (e.g., accuracy or loss) and intelligently choose the next set of hyperparameters to evaluate. It builds a surrogate model (often a Gaussian process) of the objective function and selects the next points to try based on where improvement is most likely. Unlike a grid or random search, Bayesian optimization uses past results to inform future choices, making it more sample-efficient and effective for expensive models or large search spaces.

B. Data Augmentation Methods

Data augmentation is a crucial technique in training CNNs on image samples, particularly when dealing with limited or imbalanced datasets. By artificially expanding the training set through various transformations, augmentation effectively enhances model generalization, reduces overfitting, and improves robustness to real-world variations. Common augmentation strategies include geometric transformations (e.g., rotation, scaling, flipping),

photometric adjustments (e.g., brightness, contrast variations), and more advanced techniques such as CutMix and Mixup. This section explores key data augmentation methodologies applied in CNNs and their impact on model performance.

1) *Dataset*: The full Lung and Colon Cancer Histopathological Images dataset was saved to a machine to reduce the overhead created by downloading the dataset. From this, only the Lung Histopathological Images were used for the model with 3 lung tissues: lung benign tissue (lung_aca), lung adenocarcinoma (lung_n), and lung squamous cell carcinoma (lung_scc), each containing 5000 samples for a total of 15000 samples. The dataset was further divided into three subsets: training, validation, and testing sets.

Using the TensorFlow software library for Python, the samples were loaded into a Dataset class instance. Furthermore, techniques such as caching and prefetching were used to help reduce training time and maximize GPU utilization. The use of maps also gave access to the num_parallel_calls attribute for efficient parallel processing. The parameters for these techniques (i.e., cache size, number of parallel calls) were dynamically set with the help of tf.data.AUTOTUNE.

2) *Base Hyperparameters*: To objectively evaluate the contribution of data augmentation, the hyperparameters are the same for all combinations of augmentation methods.

The following hyperparameters are set constant:

- **Batch Size** = 32
- **Input Image Size** = 64px x 64px
- **Channels** = 3
- **Input Shape** = 64 x 64 x 3
- **Epochs** = 25

The 3 data set samples were resized to 64 x 64 and the pixels were rescaled from [0, 255] to [0, 1].

3) *Training Set*: Data Augmentation is strictly applied to the training set, and the techniques used can be separated into two types: minor data augmentation (MinAug) and major data augmentation (MajAug), listed below.

MinAug:

- Random vertical and horizontal flipping
- Random rotations
- Random zooming

MajAug:

- CutMix
- MixUp
- RandAug

The CutMix and MixUp algorithms used the implementation from the example code provided by the Keras official documentation. RandAug comes from the Keras built-in preprocessing layer with the same name. MajAug's role is to reduce overfitting by introducing noise into the training set and increasing the sample count.

Instead of applying every MajAug method on the same training set, the training set is duplicated for each MajAug method tested. Furthermore, the resulting concatenated dataset of augmented samples will be shuffled first before being fed into the model for training.

4) Model and Evaluation Methods: Since this section focuses on data augmentation, the baseline model used is the same as in the Hyperparameter Optimization section. The trained model will be evaluated using the full training set, validation set, and testing set. The predictions generated using the testing set will then be represented using a confusion matrix.

C. Ensemble

A robust ensemble-based deep learning approach was developed for classifying histopathological images. Three well-established CNN architectures were employed: ResNet50, VGG16, and DenseNet121, leveraging transfer learning techniques. All three models were trained using a systematic two-phase approach and utilized real-time data augmentation to improve generalization and reduce the risk of overfitting. The predictions from these models were combined into a single ensemble model, aiming to enhance overall classification accuracy.

1) Data Preprocessing and Augmentation: The data was downloaded directly from Kaggle using the KaggleHub library. Rather than downloading and holding all images in memory (which would require significant system resources), a structured

list of file paths was created that allowed real-time loading during training and evaluation.

To efficiently manage computational resources, images were dynamically loaded and resized during model training. The dataset was split into three subsets:

- **Training Set:** used for learning optimal model parameters.
- **Validation Set:** monitored the model's performance during training, tuned the learning rate, and implemented early stopping.
- **Test Set:** reserved exclusively for the final unbiased model evaluation, ensuring metrics accurately reflect the model's generalization capabilities.

The TensorFlow.data API was used to perform image loading, resizing, and label encoding. Each image was loaded as a Dataset instance. Class labels were encoded as one-hot vectors for compatibility with the multiclass softmax classification head. To reduce training time and maximize GPU utilization, caching and prefetching techniques were employed, along with parallel processing through the `num_parallel_calls` attribute set to `tf.data.AUTOTUNE`.

Data augmentation was applied strictly to the training set to avoid overfitting and improve generalization. The augmentation techniques included:

- Random horizontal flipping
- Random rotations
- Random zooming
- Random translations
- Random contrast adjustments

These augmentations artificially increased the diversity of the training set, reducing the memorization of specific image features.

2) Model Building and Transfer Learning: Three powerful CNN architectures pretrained on ImageNet were chosen: ResNet50, VGG16, and DenseNet121. Transfer learning leveraged these pretrained models' feature extraction capabilities. For each architecture, images of size 224×224 with three color channels were provided as input, along with required preprocessing steps specific to each architecture.

Each architecture was customized by removing its original classification layers and adding new layers specific to this classification task:

- **Global Average Pooling:** Reduced spatial dimensionality and aggregated features into a single vector.
- **Batch Normalization:** Stabilized and accelerated the training process.
- **Dense Layers with Dropout Regularization:** Learned higher-level, task-specific feature representations and prevented overfitting.
- **Softmax Classification Layer:** Outputted class probabilities suited for the dataset's three classes.

Categorical cross-entropy loss was employed along with label smoothing to mitigate overconfident predictions. The Adam optimizer was chosen due to its efficiency and effectiveness in CNN training.

3) *Model Training:* The training was conducted in two distinct phases for each model to effectively leverage pretrained architectures.

Phase 1 (Feature Extraction): The pre-trained base model layers were kept frozen (weights set as non-trainable) to retain pretrained ImageNet features. Only the newly added classification head was trained with a higher learning rate. This phase allowed the model to learn task-specific features without altering the pretrained feature extraction capabilities. Each model was trained for 5 epochs using the Adam optimizer and categorical cross-entropy loss.

Phase 2 (Fine-tuning): All layers were unfrozen, enabling the model to adapt pretrained features to the specific characteristics of the histopathological images. The entire network was fine-tuned with a lower learning rate for incremental improvement in generalization. Hyperparameters were consistent across both phases.

In addition, early stopping and learning rate (LR) reduction techniques were implemented. By monitoring validation loss, training could be halted early to prevent overfitting, while LR reduction enabled refined learning and convergence.

An ensemble model was created by averaging predictions from all three independently trained models. This ensemble leveraged the collective knowledge of the three architectures, enhancing overall accuracy and robustness compared to a single model.

4) *Model Evaluation and Analysis:* After training all three CNNs, the ensemble model was evaluated on the test set to provide an unbiased assessment of real-world performance. Training and validation curves were generated to visualize model convergence and stability during training. Each architecture's training and validation loss and accuracy were plotted over epochs to facilitate diagnostic analysis of training dynamics.

Lastly, a confusion matrix was computed and visualized to provide a detailed examination of class-specific performance. This analysis identified well-classified classes and revealed potential areas of poor generalization.

V. RESULTS AND DISCUSSION

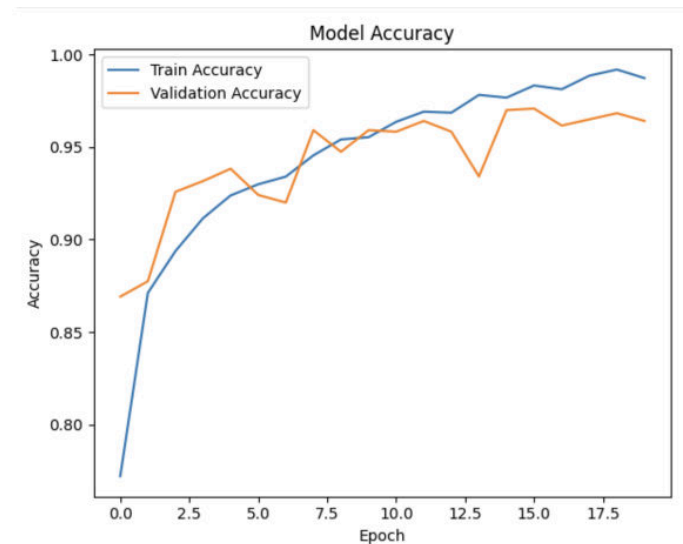


Fig. 3: Base Model Accuracy

Although the base model is simple, only containing two convolutional layers, it still provided outstanding results, scoring about 94.5% accuracy on the test set.

A. Hyperparameter Optimization

The three methods of hyperparameter optimization were compared on the basis of the average test accuracies they achieved and how much computation each consumed. Grid Search, which explored 162 combinations of hyperparameter values, achieved an average test accuracy of 93.18%. Once again, while Grid Search is an exhaustive means of

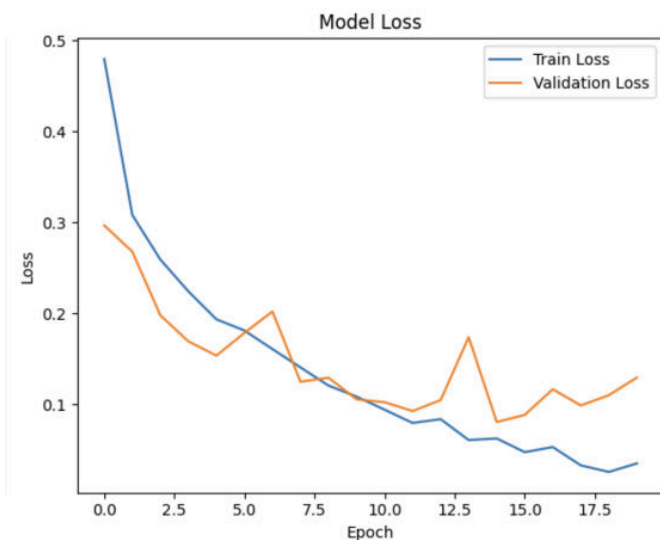


Fig. 4: Base Model Loss

searching hyperparametrically, the amount of computation and time that it consumes renders it impractical for searching hyperparameters in larger search spaces. Random Search tried 20 different random combinations from the same area and had an overall test accuracy of about 93.82% which was slightly better. It is important to note that it is a little reliant on chance and can miss the best configuration, however, its efficiency means that it is appealing to use in situations of less computational resources. Bayesian Optimization intelligently explored the hyperparameter space with a probabilistic model, and it also required significantly fewer evaluations than Grid Search. Grid Search was determined to be the leading approach after 32 iterations of testing different configurations of hyperparameters, with the highest test accuracy of 95.7%. It did this by intelligently exploring the hyperparameter space with a probabilistic model, and it also required significantly fewer evaluations than Grid Search while outperforming it as well as the Random Search. The best-performing hyperparameter configuration consisted of a filter number of 35, kernel size of 3, dense units of 128, dropout of 0.3, and a learning rate of approximately 0.001. The training logs also support that the model was improving performance throughout the epochs, as training accuracy went from 74.96% to 96.27% while validation accuracy went from 89.17% to 96.33%.

```
Epoch 8/10 7s 22ms/step - accuracy: 0.9560 - loss: 0.1181 - val_accuracy: 0.8833 - val_loss: 0.2673
Epoch 9/10 7s 22ms/step - accuracy: 0.9555 - loss: 0.1085 - val_accuracy: 0.9508 - val_loss: 0.1114
Epoch 10/10 7s 22ms/step - accuracy: 0.9627 - loss: 0.0966 - val_accuracy: 0.9633 - val_loss: 0.1012
Test accuracy with best hyperparameters: 0.9570
```

Fig. 5: Grid Search Model Results

```
Epoch 9/10 34s 101ms/step - accuracy: 0.9421 - loss: 0.1354 - val_accuracy: 0.9575 - val_loss: 0.1145
Epoch 10/10 34s 101ms/step - accuracy: 0.9513 - loss: 0.1208 - val_accuracy: 0.9692 - val_loss: 0.0873
Test accuracy with best hyperparameters: 0.9557
```

Fig. 6: Random Search Model Results

```
Epoch 8/10 10s 28ms/step - accuracy: 0.9348 - loss: 0.1698 - val_accuracy: 0.9392 - val_loss: 0.1389
Epoch 9/10 10s 29ms/step - accuracy: 0.9401 - loss: 0.1501 - val_accuracy: 0.9217 - val_loss: 0.2196
Epoch 10/10 10s 28ms/step - accuracy: 0.9464 - loss: 0.1315 - val_accuracy: 0.9550 - val_loss: 0.1063
Test accuracy with best hyperparameters: 0.9423
```

Fig. 7: Bayesian Optimization Model Results

```
[Grid Search Iteration 30] Hyperparameters: {'filters': 64, 'kernel_size': 5, 'activation': 'relu', 'dense_units': 128, 'dropout_rate': 0.3, 'learning_rate': 0.0001}
[Grid Search Iteration 30] Validation Accuracy: 0.9575
[Grid Search Iteration 31] Hyperparameters: {'filters': 64, 'kernel_size': 5, 'activation': 'relu', 'dense_units': 128, 'dropout_rate': 0.5, 'learning_rate': 0.001}
[Grid Search Iteration 31] Validation Accuracy: 0.9475
[Grid Search Iteration 32] Hyperparameters: {'filters': 64, 'kernel_size': 5, 'activation': 'relu', 'dense_units': 128, 'dropout_rate': 0.5, 'learning_rate': 0.0001}
[Grid Search Iteration 32] Validation Accuracy: 0.9358
Grid Search completed.
Best Hyperparameters: {'filters': 32, 'kernel_size': 3, 'activation': 'relu', 'dense_units': 128, 'dropout_rate': 0.3, 'learning_rate': 0.001}
Best Validation Accuracy: 0.9725
```

Fig. 8: Optimized hyperparameters after 32 Iterations of Grid Search

B. Data Augmentation

Applying data augmentation to the training set was perfectly capable of improving accuracy. As the sample count increased with each additional training set with MajAug applied, the model accuracy on the testing set also improved.

1) *Model Training Trends:* Although seeds were specified for randomized functions such as shuffling and MinAug, retraining the model multiple times yields different results each time. The best model out of 3 retrains will be looked at in this section.

Starting with the baseline model, it performed the worst as expected, with a loss of 0.4306 and an accuracy of 96.33% on the testing set. This can be attributed to having a small sample size and little to no generalization methods applied.

The model with only MinAug applied to the training set struggled to improve its validation loss

	Train Loss	Valid Loss	Test Loss	Train Acc	Valid Acc	Test Acc
No Aug	0.4439	0.4929	0.4306	79.67%	83.53%	86.33%
MinAug	0.005955	0.3714	0.3964	99.87%	91.40%	92.20%
MinAug + CutMix	0.1681	0.2017	0.2104	97.78%	92.53%	92.53%
MinAug + MixUp	0.2872	0.3335	0.3182	92.18%	88.60%	89.80%
MinAug + RandAug	0.1578	0.17	0.1603	93.32%	93.27%	93.93%
MinAug + CutMix + MixUp	0.2838	0.1218	0.09992	93.13%	95.53%	96.93%
MinAug + CutMix + RandAug	0.4803	0.2307	0.2031	76.14%	92.27%	94.00%
MinAug + MixUp + RandAug	0.0644	0.2136	0.1408	99.21%	95.07%	96.07%
MinAug + CutMix + MixUp + RandAug	0.2167	0.06393	0.06068	95.10%	97.93%	98.27%

TABLE I: Loss and Accuracy Results of all MinAug MajAug Combinations

and even deteriorated with more epochs. This indicates overfitting which was expected from training a model on a relatively small sample with high epochs. The model performed with a loss of 0.430 and accuracy of 86.33%.



Fig. 9: Loss and Accuracy trend of the MinAug model over epochs

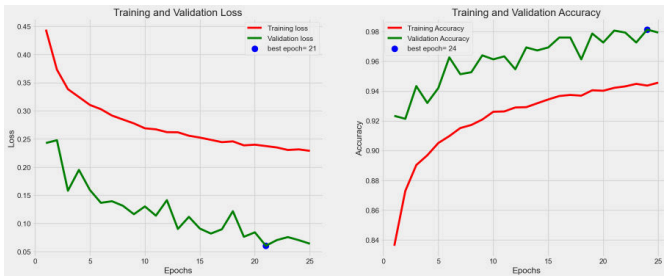


Fig. 10: Loss and Accuracy trend of the MinAug-MajAug model over epochs

Training and validation losses trend downwards further down the epoch, but most models have a much higher training loss than their validation loss. This is difficult to work out a conclusion from as on one hand, lower validation loss means the model learned difficult cases from the training set as was the intended purpose of data augmentation, resulting in a thoroughly generalized model. On the other

hand, the validation set may be mostly comprised of samples with easy predictions

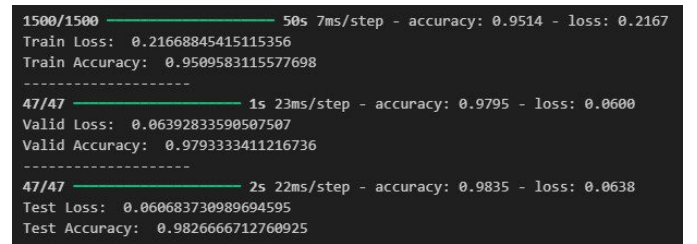


Fig. 11: Loss and Accuracy of the final MinAug-MajAug model

	precision	recall	f1-score	support
lung_aca	0.95	0.99	0.97	484
lung_n	1.00	1.00	1.00	511
lung_scc	0.99	0.95	0.97	505
accuracy			0.98	1500
macro avg	0.98	0.98	0.98	1500
weighted avg	0.98	0.98	0.98	1500

Fig. 12: Classification Report of the MinAug-MajAug model on the testing set

The best-performing combination of augmented training sets (TS) was a concatenation of TS with MinAug, TS with MinAug and CutMix, TS with MinAug and MixUp, and TS with MinAug and RandAug (MinAug-MajAug model). Since one training set has 375 batches, this results in a total of 375 x 4 = 1500 batches as the full training set. With the benefit of training on a larger sample, the model performed consistently well over the training, validation, and testing set. On the testing set, the

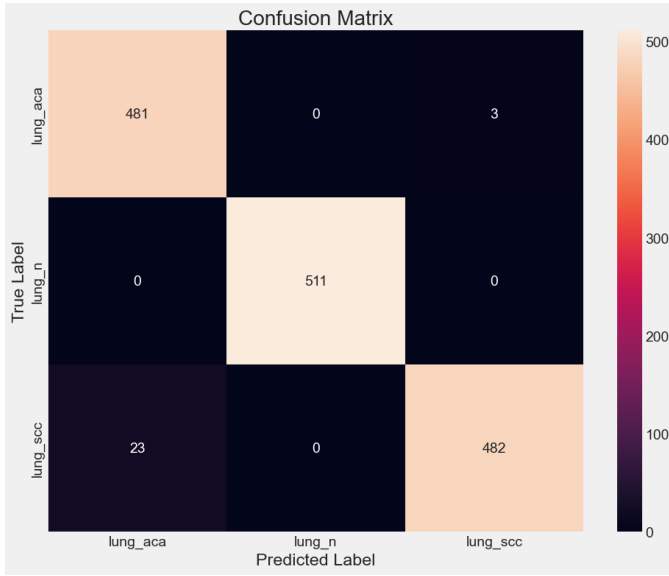


Fig. 13: Confusion Matrix of the MinAug-MajAug model on the testing set

MinAug-MajAug model had a loss of 0.061 and an accuracy of 98.27%.

Table I shows the results of the training, validation, and test loss and accuracy of every combination of MinAug and MajAug.

C. Ensemble

The ensemble model demonstrated exceptional performance by combining predictions from ResNet50, VGG16, and DenseNet121. The ensemble achieved an overall testing accuracy of 99.83% on lung histopathological images, indicating a robust and highly accurate classification system for this domain.

This outcome highlights the significant advantages of ensemble modeling, particularly for complex image recognition tasks involving subtle distinctions. Analysis of individual model performance revealed high independent accuracy, but combining the predictions into an ensemble resulted in even higher accuracy. Each model contributed unique correct predictions, effectively complementing one another. When a model struggled with subtle differences, other models compensated, thereby reducing error rates.

The high accuracy achieved by the ensemble indicates strong generalization beyond the training

dataset. Such robust performance is particularly critical in medical applications, where accurate diagnostic decisions significantly impact patient outcomes. Additionally, consistent convergence across models indicated minimal overfitting, as further supported by the analysis of training and validation curves and the confusion matrix.

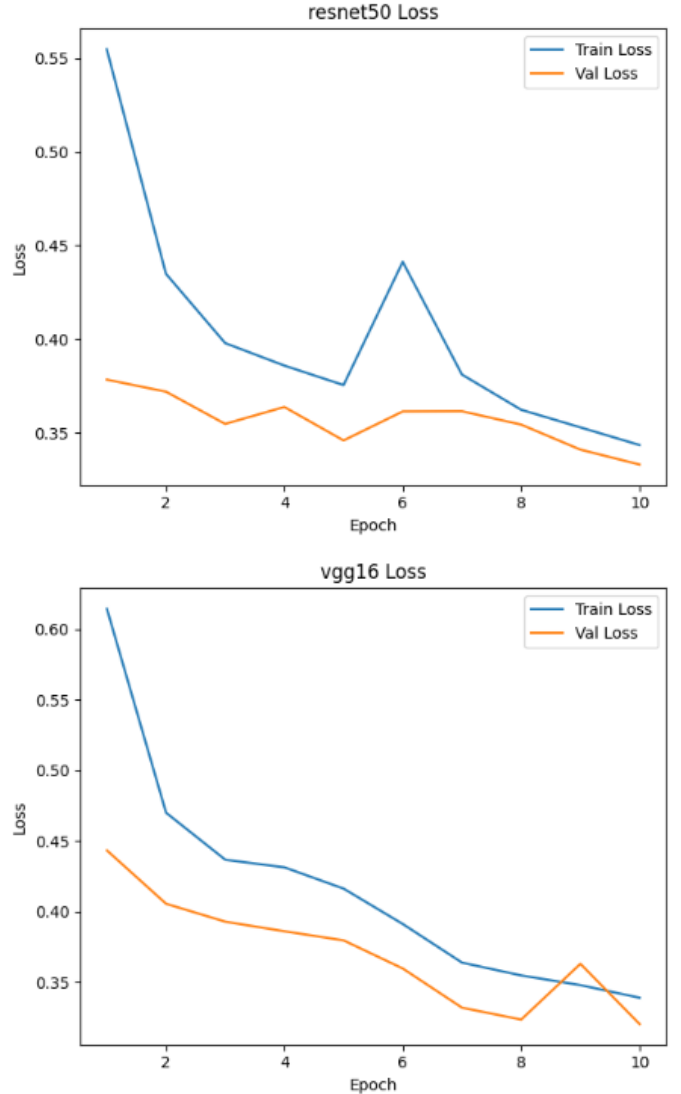


Fig. 14: ResNet50 and VGG16 loss curves.

1) *Individual Model Training*: All three models achieved high validation accuracy during training, with VGG16 and ResNet50 reaching accuracies higher than 99% by the end of the training process. In contrast, DenseNet121 did not perform as

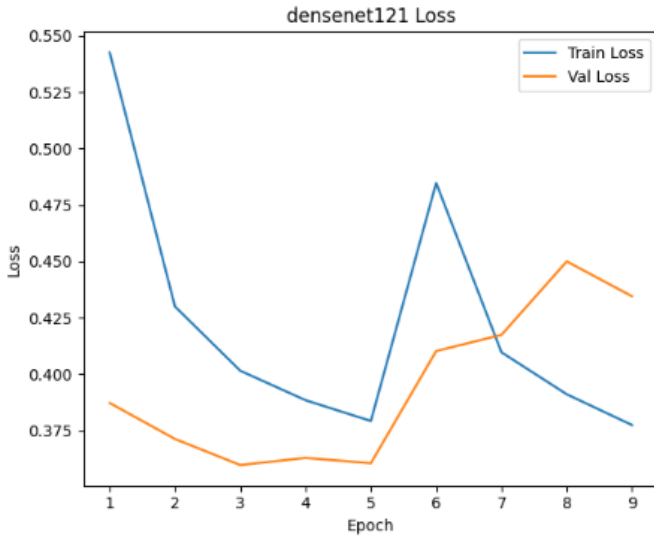


Fig. 15: DenseNet121 loss curves.

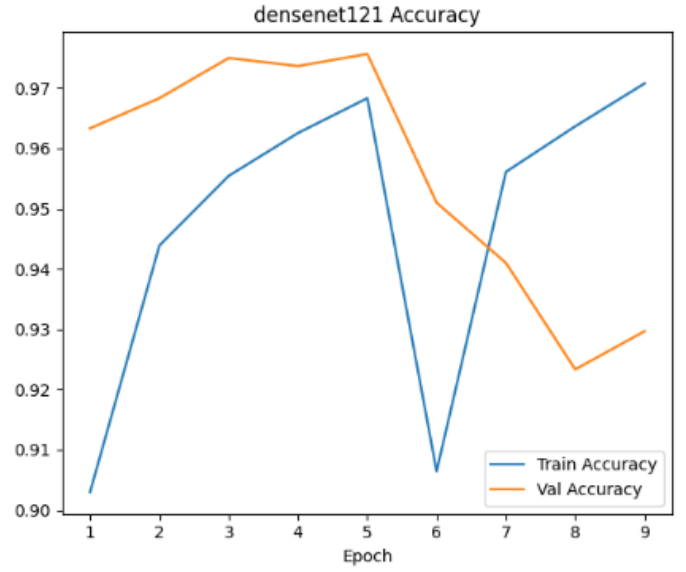


Fig. 17: Densenet121 accuracy curves.

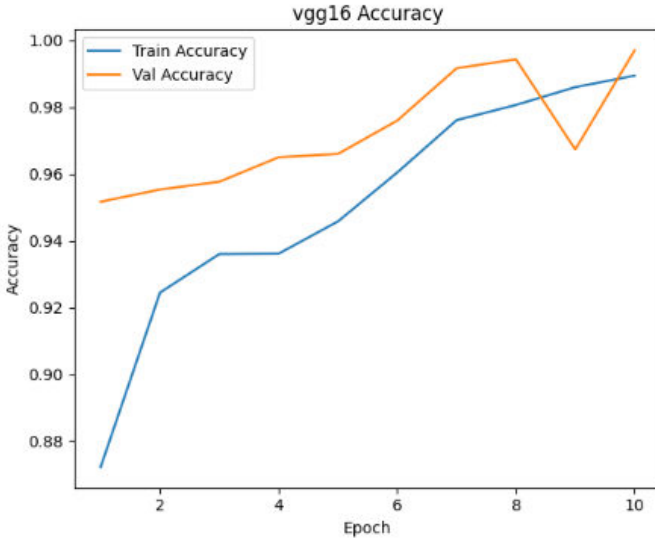


Fig. 16: VGG16 accuracy curves.

effectively, achieving just below 93% accuracy.

A detailed analysis of the training and validation curves provided insights into the learning behaviors of each ensemble model:

For validation loss, ResNet50 and VGG16 exhibited smooth downward trends, reflecting steady improvement over the training epochs (Figure 14).

DenseNet121 showed an increase in validation loss after the base model was unfrozen, indicating deteriorating performance (Figure 15).

Notably, ResNet50 and DenseNet121 experienced noticeable increases in training loss during Epoch

5, coinciding with the unfreezing of the base model layers. VGG16 did not display this behavior (Figures 14 and 15).

Accuracy curves mirrored these loss observations, with VGG16 maintaining a smooth improvement trajectory (Figure 16).

Conversely, DenseNet121 experienced decreased validation and training accuracy following the fifth epoch. Given that training accuracy recovered but validation accuracy did not, this suggested potential overfitting (Figure 17).

Early stopping was not activated during training for any of the models. However, DenseNet121 triggered a learning rate reduction during both training phases, aligning with observations of potential overfitting.

VI. FUTURE WORK

While the results were mainly positive, many points of investigation had to be overlooked due to time constraints. Similarly, there were many aspects of the methodology that could be improved in future works.

Maintaining a consistent data preprocessing pipeline would have made way for a fair comparison of the three different methodologies applied to CNN. More specifically, if the same pipeline was applied for the Hyperparameter Optimization, Data Augmentation, and Ensemble sections, then

not only would the three results be comparable to each other, but it would make it possible to combine everything into a single model that applies all three methodologies. More methodologies can then be introduced and applied to this resulting model for additional investigation.

Utilizing cross-validation techniques would help reduce skewed datasets that result in training trends as was seen in the Data Augmentation section. Furthermore, a model with minimal accuracy and loss variation after each full re-train may be a sign of a properly trained and optimized model. Achieving this makes the results easier to recreate and ensures consistent and reliable comparisons between various models.

Lastly, introducing and mixing different layers from the Keras.layers API such as Separable Conv2D and Masking layer may have interesting results. The baseline model used was extremely simple to take into account the training time. Paradoxically, shrinking the baseline model may be plausible to add more emphasis on the effects of the three methodologies used.

VII. CONCLUSION

A. Hyperparameter Optimization

This study compared three hyperparameter optimization methods Grid Search, Random Search, and Bayesian Optimization for tuning a CNN designed for lung histopathological image classification. Grid Search, which exhaustively evaluated 162 hyperparameter combinations, achieved an average test accuracy of 93.18%, but its computational demands make it impractical for larger search spaces. Random Search, sampling 20 random configurations, slightly improved performance to 93.82% while offering greater efficiency, though it risks missing the optimal setup due to its stochastic nature.

In contrast, Bayesian Optimization, which leverages a probabilistic model to intelligently explore the hyperparameter space, required significantly fewer evaluations and ultimately achieved the highest test accuracy of 95.7% after 32 iterations. The optimal configuration identified consisted of 35 filters, a kernel size of 3, 128 dense units, a dropout rate of 0.3, and a learning rate of approximately 0.001.

Future research may explore dynamic augmentation policies that adapt during training, as well as domain-specific augmentation methods tailored to specialized applications like medical imaging or autonomous driving. In conclusion, data augmentation remains an essential component for training a Convolution Neural Network, improving its accuracy and reliability in exchange for time and memory. Its ability to artificially expand the sample size also reduces the effort required for finding large datasets, and inherently increases the variation and noise, thereby increasing the generalization of the model. However, the effectiveness of augmentation depends on the dataset and task, requiring careful selection of appropriate techniques to avoid introducing unrealistic distortions.

C. Ensemble

This research evaluated the effectiveness of ensemble methods combined with various techniques aimed at enhancing generalization for histopathological image classification. By integrating predic-

tions from three established CNN architectures [7] ResNet50, VGG16, and DenseNet121, and employing a systematic two-phase training strategy supported by data augmentation, the ensemble achieved an impressive accuracy of 99.8%. This accuracy surpassed each model's individual validation performance, confirming the benefit of ensemble learning [9].

Detailed analysis revealed strong performances from ResNet50 and VGG16, each surpassing 99% validation accuracy. In contrast, DenseNet121 achieved lower accuracy (around 93%) and exhibited signs of overfitting, particularly after re-tuning. Despite this, the ensemble effectively compensated for individual model weaknesses by aggregating their class probability predictions.

Overall, the robust performance demonstrated by the ensemble model highlights its considerable potential for medical diagnostic applications, where precise classification critically impacts patient outcomes. Future research should prioritize optimizing DenseNet121 through advanced hyperparameter tuning and consider incorporating additional CNN architectures to further enhance ensemble performance.

VIII. REFERENCES

REFERENCES

- [1] S. R. Kadam, A. Srinivasan, S. K. Saurabh, P. K. Yadava, M. S. Khan, and V. K. Shrivastava, "Deep learning classification of lung cancer histology using CT images," *Sci. Rep.* vol. 11, Art. no. 4333, 2021, doi: 10.1038/s41598-021-84630-x.
- [2] A. A. Abd El-Aziz, A. A. Sewisy, and H. M. Amer, "Lung Cancer Classification in Histopathology Images Using Multiresolution Efficient Nets," *Contrast Media Mol. Imaging* vol. 2023, Art. no. 9806243, 2023, doi: 10.1155/2023/9806243.
- [3] A. K. M. F. Haque, M. M. Islam, M. S. Islam, and M. S. Islam, "Hyperparameter Tuning and Pipeline Optimization via Grid Search Method and Tree-Based AutoML in Breast Cancer Prediction," *J. Healthc. Eng.* vol. 2021, Art. no. 6615586, 2021, doi: 10.1155/2021/6615586.
- [4] S. K. Sahu, A. K. Sahu, and P. K. Sahu, "Effect of Grid Search and Hyper Parameter Tuned Pipeline with Various Classifiers and PCA for Breast Cancer Detection," *Curr. Med. Imaging* vol. 18, no. 2, pp. 171-179, 2022, doi: 10.2174/1573405617666210923142939.
- [5] Y. Zhang, Y. Li, and Y. Zhang, "A Novel MRI Diagnosis Method for Brain Tumor Classification Based on CNN and Bayesian Optimization," *BioMed Res. Int.* vol. 2022, Art. no. 9824383, 2022, doi: 10.1155/2022/9824383.
- [6] Y. Shao, C. Han, J. Wu, and S. Wang, "A selective CutMix approach improves generalizability of deep learning-based grading and risk assessment of prostate cancer," *Pathol. Inform.* vol. 15, Art. no. 100344, 2024, doi: 10.1016/j.jpi.2024.100344.

- Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "MetaMed: Few-shot medical image classification using gradient-based meta-learning," *Pattern Recognit.* vol. 120, Art. no. 108111, 2021, doi: 10.1016/j.patcog.2021.108111.
- S. Hussein, R. Gillies, K. Cao, Q. Song, and U. Bagci, "Predicting Nodule Malignancy using a CNN Ensemble Approach," *Int. J. Comput. Assist. Radiol. Surg.* vol. 13, no. 11, pp. 1819-1827, 2018, doi: 10.1007/s11548-018-1845-5.
- A. Iqbal, G. Zhu, and M. M. Fraz, "Breast Cancer Classification through Meta-Learning Ensemble Technique Using Convolution Neural Networks," *Diagnostics* vol. 13, no. 13, Art. no. 2242, 2023, doi: 10.3390/diagnostics13132242.
- M. Masud, A. K. Bairagi, A. Althaiban, and S. Alsubaie, "Explainable lung cancer classification with ensemble transfer learning of VGG16, Resnet50 and InceptionV3 using grad-cam," *BMC Med. Imaging* vol. 24, Art. no. 13, 2024, doi: 10.1186/s12880-024-01345-x.