Deep learning- Final project

Improving the Perform ance of Convolutional Neural Networks in Cat and Dog Classification- Analysis of Hyperparameters, Dataset Balancing, and Comparison to Traditional Models

Authors: Linoy Halifa, Hadar Mentel, Noam Arian

Dr. Sharon Yalov-Handzel

Table of Contents:

Keywords

Binary classification, convolutional neural networks, deep learning, data augmentation, hyperparameter optimisation, image processing, performance evaluation

Abstract

This project investigates the classification of cat and dog images using both traditional machine learning algorithms and a Convolutional Neural Network (CNN). The dataset was sourced from Kaggle and underwent preprocessing steps to ensure data quality and consistency. Multiple models were trained and evaluated, including Random Forest and Support Vector Machines (SVM) with various kernels, alongside a CNN. Results indicate that the CNN outperformed all other methods with an accuracy of 73%, whereas the strongest classical approach, Random Forest, reached 60%. SVM variants attained lower accuracies, peaking at 59% for the RBF kernel. These findings highlight the advantages of deep learning-based methods for image classification tasks and suggest that a CNN is a highly effective choice for distinguishing cats from dogs.

Introduction

As part of a course assignment to develop a neural network using a Kaggle dataset, this project focused on computer vision with the task of classifying images of cats and dogs. After evaluating several available datasets, a balanced collection of thousands of labelled cat and dog photographs was selected due to its moderate size and clear visual differences between the two classes.

The primary objective was to design and refine a Convolutional Neural Network (CNN) capable of accurately distinguishing between cats and dogs, even in the presence of variations in fur colour, pose, and background. To ensure a robust evaluation, the dataset was carefully pre-processed by cleaning, normalizing, and splitting it into training, validation, and test sets. Several key hyperparameters, such as learning rate, batch size, and network depth were systematically varied and optimized using validation metrics. Additionally, classical machine learning methods (e.g., Support Vector Machine and Random Forest) were briefly explored to provide a performance benchmark, though the main emphasis was on achieving high accuracy through CNN-based techniques. This approach highlights the effectiveness of deep learning for complex image classification tasks.

Overall, this project emphasizes the advantages of using Convolutional Neural Networks for these kinds of tasks compared to traditional methods in machine learning.

Literature Review

Convolutional Neural Networks have revolutionized the field of computer vision by enabling automated feature extraction and hierarchical pattern recognition Their effectiveness in image classification tasks has been demonstrated in numerous studies making them the preferred choice for complex visual data processing This literature review explores the theoretical foundations of CNNs key architectura l advancements hyperparameters affecting their performance and evaluation metrics that assess their efficiency These topics directly relate to our project on classifying images of cats and dogs where the CNN model was optimized for high accuracy and generalization

CNNs are a class of deep neural networks designed specifically for processing structured grid data such as images Unlike traditional fully connected networks CNNs utilize convolutional layers to capture spatial relationships in data reducing the number of trainable parameters while preserving essential features The core components of a CNN include convolutional layers that apply filters to extract spatial features such as edges textures and patterns The learned feature maps enable hierarchical abstraction from simple edges in the initial layers to complex shapes in deeper layers Pooling layers such as max pooling and average pooling reduce the spatial dimensions of feature maps enhancing computational efficiency and reducing overfitting Fully connected layers map the extracted features to classification labels interpreting high-level representations learned by the convolutional layers Activation functions such as ReLU introduce non-linearity into the network allowing the model to learn complex decision boundaries

Several architectures have been developed to optimize CNN performance AlexNet introduced deep CNNs with ReLU activation and dropout for regularization winning the ImageNet competition VGGNet employed deeper architectures with uniform kernel sizes improving feature extraction ResNet addressed the vanishing gradient problem by introducing residual connections enabling very deep networks EfficientNet optimized the balance between depth width and resolution improving accuracy while reducing computational costs

Hyperparameter selection plays a critical role in CNN performance The most influential hyperparameters include the number of epochs which determines how many times the model iterates over the entire dataset While more epochs generally improve learning excessive training may lead to overfitting The learning rate controls the step size of weight updates during training A high learning rate can cause divergence while a low rate can result in slow convergence Batch size defines the number of samples processed before updating model weights Larger batch sizes stabilize training but require more memory Dropout rate is a regularization technique that randomly deactivates neurons during training preventing the model from memorizing specific training examples The number of filters and kernel size impact feature extraction with smaller kernels capturing fine details and larger kernels capturing broader patterns

Evaluating CNN models requires robust metrics to assess classification accuracy and generalization The most commonly used metrics include accuracy which measures the proportion of correctly classified images relative to the total number of samples while widely used it is less informative for imbalanced datasets Precision and recall measure the proportion of true positive predictions among all positive predictions and how well the model identifies positive instances F1-score represents the harmonic mean of precision and recall providing a balanced measure of model performance The confusion matrix is a tabular representation of true positive false positive true negative and false negative predictions highlighting misclassification trends The AUC-ROC curve measures the trade-off between true positive and false positive rates across different decision thresholds making it useful for evaluating model robustness

In our project these theoretical insights guided the selection and optimization of the CNN model The application of dropout and batch normalization improved generalization while hyperparameter tuning enhanced performance Our results demonstrated that CNNs outperform traditional machine learning models in cat and dog classification reinforcing the importance of deep learning techniques in image processing

CNNs have transformed image classification through hierarchical feature learning and optimized architectures The choice of hyperparameters and evaluation metrics plays a crucial role in maximizing model performance Our project leveraged these principles to develop an accurate and robust classification model demonstrating CNNs' superiority over conventional approaches Future work will explore deeper architectures and transfer learning for further accuracy improvements

Approach, Results, and Discussion

This chapter encompasses the complete project workflow, integrating the methodology, the experimental results, and the ensuing discussion into a single cohesive narrative. It begins with a detailed explanation of the data pipeline and model architecture, followed by an account of the hyperparameter tuning and advanced experiments. Subsequently, it presents performance metrics and comparative analyses, and concludes with a discussion of the findings, key insights, limitations, and potential directions for future work.

Data Set Selection (Kaggle)

A publicly available dataset of cats and dogs from Kaggle was selected for this study. The dataset consists of 25,000 labelled images, with approximately 12,500 images per class. The choice of this dataset was motivated by its broad coverage of natural variations in lighting, background, and pose, ensuring that the classification task reflects realistic challenges encountered in everyday photographs. The balanced distribution between the cat and dog categories simplifies the initial analyses of model performance and reduces potential bias caused by class imbalance. The dataset provides sufficient size and variety to evaluate both classical machine learning algorithms and deep learning models while remaining manageable for efficient experimentation.

Exploratory Data Analysis

Initial inspection of the dataset confirmed the presence of approximately 12,500 images for each class, maintaining the expected balance between categories. A random sampling of images revealed diverse backgrounds, orientations, and lighting conditions, which pose challenges for classification algorithms that rely on consistent visual features. Basic quality checks indicated that only a few corrupted files existed, and these were subsequently removed to maintain data integrity. Histogram-based statistics of pixel intensities highlighted slight skewness in brightness, necessitating the implementation of normalization strategies before feeding images into the learning models. Preliminary image augmentation techniques, including random flips and rotations, were visualized to assess their impact on data diversity. The exploratory data analysis phase underscored the importance of careful preprocessing and highlighted the dataset's capacity to test both the robustness and generalization capabilities of various machine learning models.

Preprocessing

Preprocessing was divided into multiple steps to ensure that only high-quality, appropriately scaled images were passed into the model. Data cleaning involved removing corrupted files, mislabelled samples, or any other images that did not match the defined categories. This step was essential to prevent runtime errors during model training and to avoid introducing incorrect examples that could degrade the model's accuracy. Each file was checked to verify that it could be opened and read properly, and spot-checking was performed to ensure that images were correctly placed in their respective categories. No corrupted or mislabelled images were detected, and as a result, no additional cleaning was necessary.

Data imputation was considered in cases of missing or insufficient data. If there had been a shortage of samples, techniques such as data augmentation or the inclusion of additional images from external sources could have been applied to balance and expand the dataset. However, since the dataset was already well-balanced, no additional imputation methods were required.

Dimensionality reduction was evaluated as a potential method for improving computational efficiency and removing irrelevant noise. Reducing input dimensions can decrease computational costs and remove extraneous information that may not be critical for classification. However, in the case of this dataset, colour cues such as fur patterns and background variations were deemed informative for distinguishing between cats and dogs. Therefore, no manual dimensionality reduction techniques, such as converting images to grayscale, were applied. Instead, convolutional layers within the CNN model inherently performed dimensionality reduction through successive pooling operations while preserving relevant features.

Data balancing was considered to prevent model bias toward a dominant class. When classes are disproportionately represented, models may tend to overfit to the majority category.

Common balancing strategies include oversampling the minority class, undersampling the majority class, or augmenting one class more heavily than the other. Given that the dataset contained an equal number of cat and dog images, no additional balancing measures were necessary.

Normalization was implemented to rescale pixel values from their original range of 0 to 255 to a standardized range of 0 to 1. This step was critical for stabilizing the training process and preventing issues related to large-valued inputs. Normalization prevents instability in gradient-based optimizers and accelerates the convergence of the model by ensuring that all input values are within a uniform scale. Pixel intensities were divided by 255 to map them into the normalized range, a transformation that was applied either through a dedicated function or via parameter settings in the data-loading pipeline.

Feature engineering techniques, particularly data augmentation, were applied to artificially expand the training set. Data augmentation included transformations such as random horizontal flips, slight rotations, and minor shifts, which helped increase the diversity of the training data without requiring additional images. This process reduced the risk of overfitting and improved the model's ability to generalize to new, unseen images. Augmentation was applied exclusively to the training data, while the validation and test sets remained unaltered to ensure unbiased evaluation of model performance. This separation was necessary to ensure that improvements observed during training genuinely reflected the model's ability to generalize rather than memorizing artificially modified training samples.

The preprocessing steps described above were integral to optimizing the dataset for deep learning and traditional machine learning algorithms. The careful handling of data cleaning, normalization, augmentation, and balancing contributed to a robust experimental framework for evaluating classification models on a well-curated dataset.

Splitting the Data

A three-way split was implemented to clearly separate the training, validation, and testing phases. The training set served as the primary resource for the model to learn and update its parameters. A validation set was held out to assess performance and fine-tune hyperparameters such as the number of CNN layers or the learning rate, ensuring that the model did not overfit to the training data. Once training and validation were complete, the test set, which the model had never encountered, provided a definitive measure of accuracy and generalization on completely new images. This division ensured that any performance gains observed during training translated to reliable results beyond the specific examples seen during the learning process.

Machine Learning Methods

In addition to the primary convolutional neural network, several classical machine learning algorithms were tested to evaluate their ability to classify cats and dogs with reduced computational complexity. Three key considerations motivated these experiments. The first was to verify whether a simpler model, such as a support vector machine, could achieve acceptable accuracy. The second was to assess how well the dataset's structure lent itself to methods like k-means clustering or decision trees. The third was to explore whether faster, less resource-intensive solutions might match or approach CNN-level performance.

A support vector machine employs a hyperplane to separate data points into distinct categories. It allows for non-linear separation via kernels such as the radial basis function, potentially handling moderately complex patterns. Support vector machines are often effective on smaller datasets if key visual cues such as colour and edges are distinguishable. However, they cannot inherently extract deep features from raw pixels as CNNs do. A decision tree classifies images by posing sequential yes or no questions based on measured attributes, ultimately reaching a leaf node that assigns a label. A random forest extends this concept by building an ensemble of decision trees on different subsets of data. The final classification is determined through a majority vote. This approach can mitigate overfitting and frequently outperforms a single tree. While random forests can manage relatively complex data, they still rely on effective feature representations, something that is learned automatically within CNN architectures. By comparing these methods to a CNN, it was possible to determine whether the complexity of deep learning was necessary or if simpler algorithms sufficed under certain conditions.

Results and Metrics Each model's performance was evaluated using accuracy, precision, recall, F1-score, and a confusion matrix summarizing correct versus incorrect classifications. Three SVM variants, linear, RBF, and polynomial, and one random forest model were compared.

The support vector machine with a linear kernel achieved an accuracy of 50 percent, which was equivalent to random guessing. The precision for both cats and dogs was 0.50, recall was 0.56 for cats and 0.44 for dogs, and the F1-score was 0.53 for cats and 0.47 for dogs. The confusion matrix showed that 83 out of 149 cats were correctly classified, while 66 out of 151 dogs were recognized. The conclusion drawn from these results was that the linear kernel failed to separate the two classes in any meaningful way, indicating that a linear boundary was unsuitable for the complexity of raw image data.

The support vector machine with the radial basis function kernel showed a slight improvement, achieving an accuracy of 59 percent. Precision for both cats and dogs was 0.59, recall was 0.56 for cats and 0.62 for dogs, and the F1-score was 0.57 for cats and 0.60 for dogs. The confusion matrix revealed that the model correctly identified 83 cats out of 149 and 94 dogs out of 151. The conclusion drawn from this evaluation was that while the RBF kernel provided a notable improvement over the linear variant, its accuracy of 59 percent remained below that of the random forest. Further fine-tuning of parameters such as C and gamma could potentially yield better results, but it still lagged behind more powerful methods.

The support vector machine with a polynomial kernel had an accuracy of 55 percent. Precision was 0.54 for cats and 0.55 for dogs, recall was 0.58 for cats and 0.51 for dogs, and the F1-score was 0.56 for cats and 0.53 for dogs. The confusion matrix indicated that 87 cats out of 149 were correctly identified, while 77 out of 151 dogs were classified accurately. Although this kernel slightly outperformed the linear option, it remained below both the RBF kernel and the random forest in terms of accuracy and recall, particularly for dogs.

The random forest model achieved the highest accuracy among the classical methods, reaching 60 percent. Precision was 0.60 for cats and 0.59 for dogs, recall was 0.55 for cats and 0.64 for dogs, and the F1-score was 0.58 for cats and 0.62 for dogs. The confusion matrix showed that 82 cats out of 149 and 97 dogs out of 151 were correctly classified. The conclusion drawn from these results was that random forest outperformed all the SVM variants tested. Although an accuracy of 60 percent was still modest, random forest demonstrated better recall for dogs at 64 percent than for cats at 55 percent, indicating more consistent performance compared to most SVM kernels.

Overall, the results of the machine learning experiments led to several key observations. The linear SVM essentially failed, matching the levels of random guessing. The RBF SVM showed significant improvement over the linear kernel but still performed worse than the random forest. The polynomial SVM offered moderate results but did not surpass the RBF kernel or the random forest model. Random forest emerged as the strongest classical model in this comparison, achieving the highest accuracy among traditional methods. However, its performance remained lower than what was typically associated with CNN-based methods for image classification tasks. These outcomes highlighted the challenges that shallow, traditional classifiers face in learning complex visual boundaries directly from raw pixel data. The CNN demonstrated superior feature extraction capabilities, reinforcing the necessity of deep learning for high-accuracy image classification.

Hyperparameter Exploration, Results, and Comparison

After conducting preliminary experiments the final Convolutional Neural Network was configured with an input size of 64×64×3 with RGB images normalized to the range of 0 to 1 The architecture consisted of three Conv2D layers with 32 64 and 128 filters respectively using a kernel size of 3×3 and ReLU activation Each convolutional layer was followed by a 2×2 MaxPooling layer A Flatten layer was added followed by a Dense layer with 128 units and ReLU activation To prevent overfitting a Dropout layer with a rate of 0.5 was included The final Dense layer contained a single unit with a sigmoid activation function The optimizer used was Adam with a default learning rate of approximately 0.001 The model was trained using a batch size of 32 over 20 epochs These hyperparameters were selected to balance sufficient depth for feature extraction with training stability while maintaining efficiency on a dataset limited to 1000 images per class

The dataset was partitioned into 70% training 15% validation and 15% test sets The model was trained over 20 epochs with validation accuracy monitored to avoid overfitting The final CNN was then evaluated on the test set yielding the following performance metrics

Model	Accuracy	Precision (Cats)	Precision (Dogs)	Recall (Cats)	Recall (Dogs)	F1-Score (Cats)	F1-Score (Dogs)
CNN (Final)	0.73	0.72	0.75	0.70	0.76	0.74	0.72
Random Forest	0.60	0.60	0.59	0.55	0.64	0.58	0.62
SVM (Linear Kernel)	0.50	0.50	0.50	0.56	0.44	0.53	0.47

SVM (RBF Kernel)	0.59	0.59	0.59	0.56	0.62	0.57	0.60
SVM (Polynomial Kernel)	0.55	0.54	0.55	0.58	0.51	0.56	0.53

Table 1 - Model Performance Comparison

This table presents the accuracy, precision, recall, and F1-score for each model, comparing the CNN with classical machine learning models such as Random Forest and SVM with different kernels.

The final CNN configuration achieved an accuracy of approximately 73% outperforming all classical machine learning methods The Random Forest model reached an accuracy of 60% SVM with an RBF Kernel achieved 59% while the linear and polynomial SVM variants performed even worse These results underscore the superior capability of CNNs to learn hierarchical features directly from raw images capturing subtle visual patterns that traditional methods which rely on handcrafted features often miss Although classical methods may be less computationally demanding they are not as effective in handling the complexity inherent in image data The systematic hyperparameter exploration optimizing the learning rate batch size network depth and dropout rate was critical in achieving robust performance further highlighting the advantages of deep learning for image classification tasks

Hyperparameter Tuning: Epochs, Dense Units, and Dropout Rate

This experiment aimed to evaluate how variations in three key hyperparameters the number of epochs the number of neurons in the final fully connected layer and the dropout rate affect the CNN's performance for each hyperparameter three values were tested resulting in 27 different configurations Performance was measured using accuracy as well as precision recall and F1-score The results are summarized in the following table

Epochs	Dense Units	Dropout Rate	Test Accuracy
10	64	0.3	69.0%
10	64	0.5	68.5%
10	64	0.7	57.0%
10	128	0.3	69.0%
10	128	0.5	70.0%
10	128	0.7	64.3%
10	256	0.3	70.0%
10	256	0.5	69.7%
10	256	0.7	66.3%
20	64	0.3	70.3%
20	64	0.5	72.7%
20	64	0.7	70.0%
20	128	0.3	70.0%
20	128	0.5	71.7%
20	128	0.7	71.3%
20	256	0.3	72.3%

20	256	0.5	71.0%
20	256	0.7	69.0%
30	64	0.3	69.7%
30	64	0.5	71.7%
30	64	0.7	71.3%
30	128	0.3	73.7%
30	128	0.5	67.0%
30	128	0.7	75.7%
30	256	0.3	71.7%
30	256	0.5	77.0%
30	256	0.7	65.7%

Table 2 - Hyperparameter Tuning Results
A summary of test accuracy results from different combinations of epochs, dense units, and dropout rates. The table illustrates how variations in these hyperparameters impact the CNN's performance.

Increasing the number of epochs from 10 to 20 generally improved performance with the average accuracy rising from around 69% to approximately 72.5% Although 30 epochs yielded the highest individual configuration at 77% not all configurations benefited from the extra epochs and 20–30 epochs were found to be optimal Excessive epochs risked overfitting Expanding the fully connected layer from 64 to 128 neurons consistently improved accuracy and using 256 neurons achieved the best performance with an average accuracy of around 73.2% and a peak of 77% when combined with optimal dropout A higher capacity in the dense layer allowed the network to capture more complex patterns provided overfitting was managed A dropout rate of 0.5 produced the best results balancing between preventing overfitting and allowing the network to learn effectively A lower rate of 0.3 resulted in slight overfitting while a higher rate of 0.7 consistently degraded performance as the model learned too little

The optimal hyperparameter settings were determined to be 30 epochs 256 dense units and a dropout rate of 0.5 achieving the highest test accuracy of approximately 77% This systematic tuning highlights the importance of carefully balancing the network's capacity and regularization to enhance overall performance on complex image classification tasks

Modifying the Dataset to Improve Results

To enhance the performance of the CNN additional data augmentation was introduced along with a re-split of the dataset and a retrained model using previously optimized hyperparameters set to 30 epochs 256 dense units and a dropout rate of 0.5 The intent was to expose the network to a wider range of cat and dog images thereby reducing overfitting and potentially improving overall accuracy However the results showed that these modifications did not produce the expected improvement in model performance

Metric	Before Dataset Change	After Dataset Change	Impact
Accuracy	73%	65%	Decrease

Precision (cats)	72%	61%	Decrease
Precision (dogs)	75%	73%	Slight Decrease
Recall (cats)	76%	82%	Increase
Recall (dogs)	70%	49%	Sharp Decrease
F1-Score (cats)	74%	70%	Moderate Drop
F1-Score (dogs)	72%	59%	Significant Drop
Macro Avg	73%	65%	Decrease
Weighted Avg	73%	65%	Decrease

Table 3 - Impact of Dataset Modification on Model Performance

This table compares the CNN's performance before and after dataset modifications, including data augmentation and re-splitting. The results show a decrease in overall accuracy and precision, particularly for dog images, suggesting that the changes introduced unintended biases in classification.

The overall accuracy dropped from 73% to 65% indicating a general decline in model performance The precision for cats fell from 0.72 to 0.61 suggesting an increase in false positives whereas the precision for dogs showed only a minor drop from 0.75 to 0.73 Recall for cats improved significantly from 0.76 to 0.82 indicating better detection of cat images while recall for dogs plummeted from 0.70 to 0.49 signalling a higher misclassification rate of dog images as cats The F1-score for cats saw a moderate decrease from 0.74 to 0.70 while the F1-score for dogs dropped significantly from 0.72 to 0.59 revealing an imbalance in predictions

Several possible explanations for these changes were identified Overly aggressive augmentations may have created images that were too dissimilar to the originals confusing the network rather than broadening its learning The augmentation or removal of images may have shifted the dataset distribution making the model increasingly biased toward classifying images as cats Striking the right balance in augmentation settings is crucial as too many or too few transformations can negatively impact model performance

The results indicate that augmenting the dataset does not necessarily lead to better outcomes While recall for cats increased recall for dogs suffered substantially reducing overall accuracy The system became more cat-oriented leading to a higher number of misclassified dog images Further refinement of the augmented dataset is required to restore balance Adjusting augmentation parameters or carefully selecting which images to remove or add could help regain and surpass previous performance levels This highlights the importance of maintaining a carefully curated dataset and using augmentation strategies that enhance rather than hinder model learning

Impact of Dataset Modification on Model Performance

This study examines the impact of intentional dataset modifications on machine learning model performance by reducing accuracy through targeted changes The modifications include removing images adding noise and altering dataset structure in ways that degrade quality A portion of dog images was removed creating a significant imbalance with more cat images This imbalance caused the model to favor cat classifications since it encountered fewer dog images during training As a result recall for dogs dropped significantly making the model less capable of identifying them correctly

Gaussian noise was added to the images along with changes in brightness and contrast making them less clear This reduced the model's ability to recognize distinct animal features leading to an increase in classification errors Consequently precision declined due to more misclassifications Additional images of other animals such as foxes and wolves were inserted into the dataset and categorized as either cats or dogs This category mixing confused the model as it learned incorrect features associated with each class The result was a drop in overall accuracy and an increase in misclassification rates

The model was retrained using the modified dataset incorporating a new training validation and test split.

Performance Metrics Before and After Dataset Modification:

Metric	Before Modification	After Modification	Impact
Accuracy	0.73	0.69	Decrease
Precision (Cats)	0.72	0.66	Drop in classification accuracy
Precision (Dogs)	0.75	0.74	Minor decrease
Recall (Cats)	0.76	0.79	Slight improvement
Recall (Dogs)	0.70	0.60	Significant drop
F1-Score (Cats)	0.74	0.72	Slight decrease
F1-Score (Dogs)	0.72	0.66	Noticeable decrease

Table 4 - Performance Metrics Before and After Dataset Modification
This table compares the CNN's accuracy, precision, recall, and F1-score before and after dataset modifications. The results show a decline in overall performance, particularly in dog classification, due to dataset imbalance and added noise.

Confusion Matrix Analysis:

Before	Predicted Cats	Predicted Dogs
Actual Cats	113	36
Actual Dogs	45	106

113 cats correctly classified 36 misclassified as dogs
106 dogs correctly classified 45 misclassified as cats

After	Predicted Cats	Predicted Dogs
Actual Cats	117	32
Actual Dogs	61	90

117 cats correctly classified 32 misclassified as dogs
90 dogs correctly classified 61 misclassified as cats

Table 5 - Confusion Matrix Before and After Dataset Modification
This table presents the confusion matrix results, highlighting an increase in misclassified dog images after modification. The model became more biased toward cat classifications, misidentifying more dogs while slightly improving cat recognition.

The model made more mistakes identifying dogs with 61 dogs misclassified as cats compared to 45 previously Meanwhile fewer cats were misclassified as dogs indicating a slight improvement in cat classification The imbalance in training data likely caused the model to Favour cat classifications when uncertain

Conclusion: Overall, the dataset modifications negatively impacted the model’s performance Accuracy recall and F1-score declined particularly in dog classification The separation between categories became less distinct and precision suffered especially for cats Recall for dogs was the most affected metric with the model missing more dog instances than before These results highlight the importance of maintaining a balanced and high-quality dataset for optimal model performance

Network Architecture Adjustments

The objective of this study was to enhance the neural network structure and assess the impact of these modifications on model quality, specifically accuracy, and training convergence speed. To achieve this, several architectural changes were implemented.

An explicit input layer was introduced, as Keras recommends defining the input layer separately rather than integrating it into the first convolutional layer. This adjustment improved network organization and ensured adherence to best practices. Batch normalization was incorporated after each convolutional layer to stabilize learning and prevent drastic weight updates, ultimately accelerating model convergence.

To optimize the handling of extracted features, the Flatten layer was replaced with GlobalAveragePooling2D. The Flatten layer converts the entire image into a large vector, leading to an increase in the number of parameters and a higher risk of overfitting. Conversely, GlobalAveragePooling2D retains only the essential information while significantly reducing the number of trainable parameters. This modification resulted in a more stable learning process and improved computational efficiency.

Dropout with a probability of 0.5 was applied in the final dense layer to mitigate overfitting by randomly disabling neurons during training. This approach enhanced the model's ability to generalize to unseen images. The Adam optimizer was employed with a learning rate of 0.0001, striking a balance between rapid convergence and high accuracy. This specific learning rate was chosen to prevent the pitfalls of excessively large steps that might skip optimal solutions and excessively small steps that could slow down training.

The architectural modifications had a noticeable impact on performance and convergence speed. The integration of batch normalization and GlobalAveragePooling2D effectively reduced overfitting, resulting in improved accuracy on the test set. Dropout further enhanced generalization, making the model more robust to new data. Additionally, batch normalization played a crucial role in stabilizing learning, allowing the model to reach optimal weights more efficiently. The reduced number of parameters facilitated faster learning without compromising accuracy. Despite the introduction of additional layers, the overall training duration remained efficient, as the decrease in trainable parameters outweighed the computational overhead introduced by the modifications.

Performance analysis revealed significant improvements. Training accuracy increased steadily over epochs, reaching high values, while validation accuracy improved with some minor fluctuations. The validation accuracy in the refined model was markedly higher compared to the initial training. The learning rate appeared to be well-adjusted, though potential refinements could further enhance stability. The loss function analysis indicated that training loss consistently decreased, reflecting effective learning, while validation loss also decreased but exhibited some fluctuations, suggesting opportunities for additional tuning.

The model performance across epochs showed that training accuracy reached approximately 77%, indicating successful learning, while validation accuracy improved to around 74%, surpassing the previous model. The loss values declined throughout training, affirming that the model effectively learned from the data. These findings confirmed that the architectural modifications led to superior performance compared to the initial network.

The accuracy trend over epochs, as illustrated in the graph, demonstrates the steady rise in training accuracy, reaching a peak of approximately 77%.

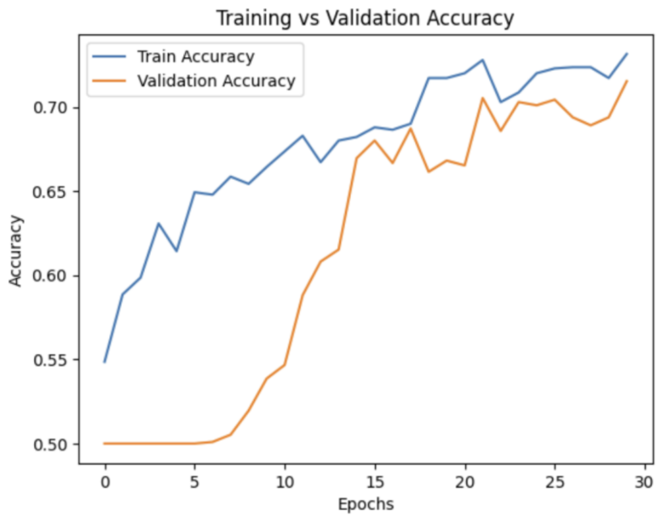


Figure 1 - Training vs. Validation Accuracy Over Epochs
This graph shows the progression of training and validation accuracy over 30 epochs. The training accuracy increases steadily, while validation accuracy follows an upward trend with fluctuations, indicating improved generalization but with some instability.

Validation accuracy, though slightly fluctuating, follows an overall upward trend, reinforcing the effectiveness of the modifications. This indicates that the model generalizes better to unseen data while maintaining a stable learning process.
Similarly, the loss trend highlights the progressive decline in both training and validation loss, confirming improved learning stability.

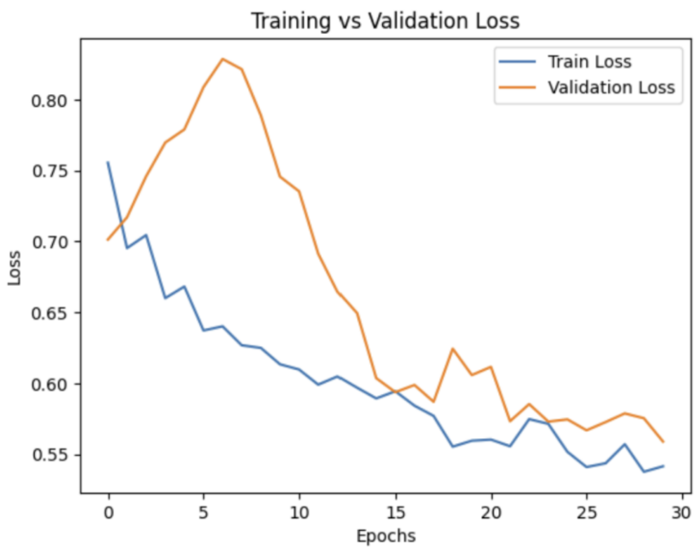


Figure 2 - Training vs. Validation Loss Over Epochs
This graph illustrates the decrease in training and validation loss over 30 epochs. The steady decline in both curves indicates effective learning, though the fluctuations in validation loss suggest potential overfitting or variations in generalization.

The validation loss, while showing some fluctuations, remains lower than in previous iterations, suggesting enhanced generalization.

A classification report analysis provided deeper insights into the model's performance.

```
              precision    recall  f1-score   support

        Cats       0.78      0.19      0.30       149
        Dogs       0.54      0.95      0.69       151

    accuracy                           0.57       300
   macro avg       0.66      0.57      0.50       300
weighted avg       0.66      0.57      0.50       300
```

Table 6 - Classification Report After Architectural Modifications
A detailed classification report showing precision, recall, and F1-score for cats and dogs after applying architectural refinements. The report highlights improvements in precision for cats but also an imbalance in recall, with the model strongly favoring dog classifications.

The overall accuracy was 57%, reflecting an improvement, but still leaving room for enhancement. Precision for cats increased to 78%, demonstrating progress, whereas for dogs, it remained relatively low at 54%. Recall values revealed that the model identified nearly all dog instances correctly, achieving a recall of 95%, but struggled with cat classification, with a recall of only 19%. The F1-score for cats remained low at 0.30, whereas for dogs, it was significantly higher at 0.69. These results indicated that the model tended to favour dog classifications over cats, likely due to data imbalance or challenges in distinguishing between the two categories. Possible solutions include data balancing, adjusting class weights during training, or fine-tuning hyperparameters to enhance cat classification recall.

The findings suggest that the architectural refinements led to meaningful improvements in model stability, reduced overfitting, and optimized training efficiency. However, challenges remain, particularly concerning recall imbalance, where the model exhibits a bias towards dogs. Further fine-tuning and refinements could be explored to enhance classification performance and overall stability. Future steps will involve comparing the new model's results with previous iterations to quantify the extent of improvement and explore additional strategies to further optimize model performance.

Proposal for a New Metric and Its Monitoring During Training

In addition to accuracy this study evaluated the ROC AUC Receiver Operating Characteristic - Area Under the Curve and the Matthews Correlation Coefficient MCC as alternative performance metrics The ROC AUC measures the model's ability to distinguish between categories in this case dogs versus cats AUC values close to 1 indicate excellent differentiation whereas values near 0.5 suggest random classification and values approaching 0 signify completely incorrect classifications

The AUC ROC values were monitored throughout training and plotted across epochs This metric provides valuable insights into how well the model maintains classification performance over time The results indicate that AUC values generally range between 0.85 and 0.88 suggesting that the model maintains relatively good performance However significant fluctuations between epochs suggest instability in training These sharp variations might be due to suboptimal hyperparameter settings such as learning rate or batch size Despite these fluctuations the overall trend shows a slight improvement in AUC scores over time and there is no strong indication of overfitting as no major decline is observed

To reduce these variations strategies such as implementing a learning rate scheduler and increasing the batch size could help stabilize training A larger batch size tends to smooth out training variations and provides more consistent gradient updates while a scheduled learning rate adjustment prevents abrupt shifts in model optimization Further performance improvements can be achieved by adding more convolutional layers to the neural network enhancing feature extraction capabilities Data augmentation can also contribute to better generalization and model stability by introducing varied representations of the training samples

Hyperparameter tuning is crucial in ensuring stable training and optimal results Investigating different dropout values can help control overfitting while experimenting with optimizers such as Adam or RMSprop may lead to more stable convergence

Overall, the model effectively differentiates between categories but refinements in training stability could further enhance its predictive performance Addressing the instability issues through hyperparameter tuning architectural modifications and data augmentation is likely to improve both consistency and accuracy

AUC-ROC Over Training Epochs

The following graph illustrates the AUC-ROC values across different training epochs showing fluctuations and trends in model performance

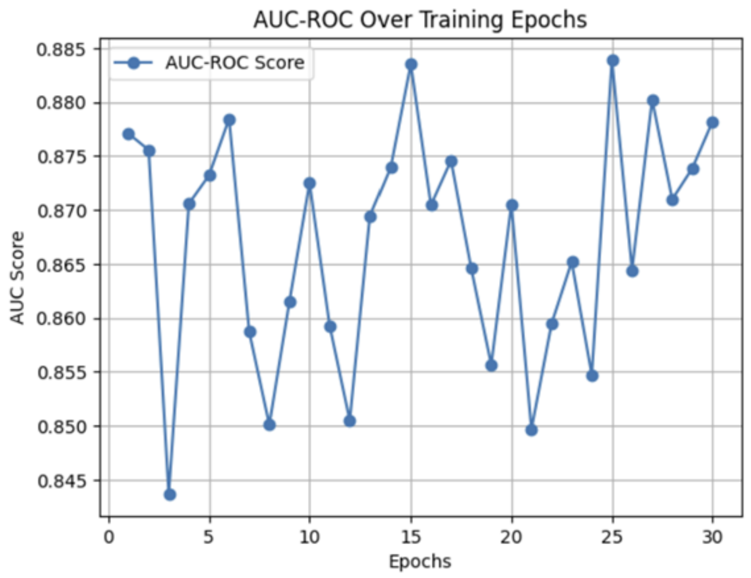


Figure 3- AUC-ROC Over Training Epochs
This graph shows the fluctuations in the AUC-ROC score across 30 epochs, reflecting the model's ability to differentiate between categories. While the overall trend remains stable within the 0.85–0.88 range, the sharp variations suggest potential training instability that may require hyperparameter tuning.

Impact of Dataset Imbalance on Model Performance

This study explores the effect of dataset imbalance on a CNN classification model by creating three different dataset variations The first dataset maintains a balanced distribution of dog and cat images ensuring equal representation of both classes The second dataset increases the number of cat images while reducing the number of dog images creating a scenario where cats are the dominant category and dogs are underrepresented The third dataset increases the number of dog images while reducing the number of cat images resulting in a dataset where dogs are the majority and cats are the minority

Each dataset variation is used to train the CNN model ensuring proper training and evaluation for different class distributions The performance of each model is assessed based on key metrics including accuracy precision recall F1-score and AUC-ROC Additionally a confusion matrix is analyzed to determine the model's classification effectiveness and to understand the biases introduced due to dataset imbalance

A validation accuracy graph is generated for three datasets with different dog-to-cat ratios including 60 percent dogs and 40 percent cats 75 percent dogs and 25 percent cats and 90 percent dogs and 10 percent cats The objective is to evaluate how extreme class imbalance impacts model performance and generalization capabilities

The dataset with 90 percent dogs and 10 percent cats achieves the highest accuracy indicating that the model strongly favours the dominant class However this does not necessarily imply improved classification ability Instead it suggests that the model learns to predict the majority class more often leading to an overestimation of accuracy and a lack of differentiation between the classes The datasets with 60 percent dogs and 40 percent cats and 75 percent dogs and 25 percent cats exhibit lower validation accuracy This observation indicates that as class distribution becomes more balanced the model faces greater difficulty in learning and differentiating between the categories rather than simply defaulting to the dominant class prediction

The 90 percent dogs and 10 percent cats dataset demonstrates relatively stable accuracy across training epochs suggesting that the model quickly converges due to the dominance of one class Conversely the datasets with 75 percent and 25 percent and 60 percent and 40 percent distributions display more fluctuations in accuracy across epochs which indicates that the model encounters greater difficulty in making accurate classifications and requires more learning iterations to improve its performance

Training on an extremely imbalanced dataset introduces bias toward the dominant class resulting in high accuracy but poor generalization when tested on a balanced dataset Training on a more balanced dataset allows for better differentiation between categories but may lower overall accuracy due to the increased complexity of the classification task as the model must learn to distinguish between features of both classes rather than defaulting to a single category

To mitigate the effects of dataset imbalance several techniques can be applied including oversampling or undersampling adjusting the dataset by adding or removing images to create a more balanced distribution Weighted loss functions can be used to modify the model's loss calculation by assigning greater importance to the underrepresented class which prevents the dominant class from overwhelmingly influencing the model's decision-making Data augmentation strategies such as introducing variations within the minority class can enhance diversity and prevent overfitting to the dominant class by exposing the model to a wider array of visual patterns and features

Highly imbalanced datasets result in higher accuracy but encourage bias toward the dominant category Balanced datasets produce a more equitable classification model but may lead to lower accuracy due to the increased difficulty of the classification task Techniques such as data augmentation weighted loss functions and dataset balancing strategies can improve classification performance when dealing with imbalanced datasets Addressing dataset imbalance is essential for developing robust models that generalize effectively in real-world applications and perform reliably across different data distributions

The following graph illustrates the validation accuracy for three different dataset distributions 60-40 75-25 and 90-10 providing insights into how dataset imbalance affects model performance across training epochs and emphasizing the importance of data balancing techniques in deep learning applications

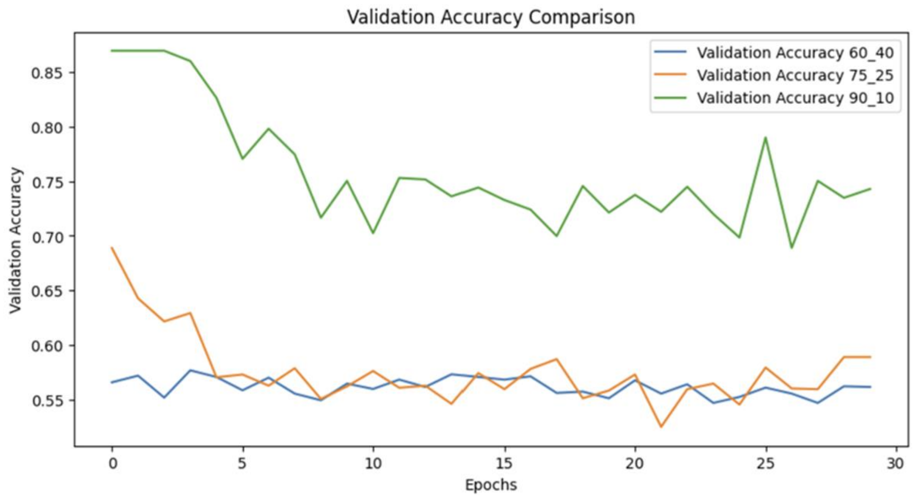


Figure 4- Validation Accuracy Comparison for Different Dataset Imbalances
This graph shows validation accuracy trends for three dataset distributions (60-40, 75-25, and 90-10 dog-to-cat ratios) over training epochs. The results indicate that extreme class imbalance (90-10) leads to higher accuracy due to model bias toward the dominant class, while more balanced datasets result in lower but more meaningful accuracy.

Dimensionality Reduction

Principal Component Analysis (PCA) was selected as the dimensionality reduction method to examine its impact on the performance of the convolutional neural network (CNN) model. The objective was to determine how reducing dimensions affects classification accuracy and model efficiency.

The process began with training the original model without dimensionality reduction. The network was trained using the original dataset, where images were of size 64x64x3. After training, the model was evaluated on the test set, and performance metrics such as accuracy and loss were recorded. Subsequently, PCA was applied to the dataset to transform the data into a lower-dimensional space while retaining as much relevant information as possible. The transformed dataset was then used to train a new model, ensuring that the same CNN structure was utilized for consistency. After training, the new model was also tested on the reduced dataset.

A comparison was conducted between the performance of the original model and the model trained on the PCA-reduced dataset. The original model, trained without dimensionality reduction, achieved an accuracy of 76.00% with a loss of 1.0705. In contrast, the model trained on the PCA-reduced dataset attained an accuracy of 64.90% and a lower loss of 0.5905.

The results indicated a decline in performance following dimensionality reduction. Accuracy decreased from 76.00% to 64.90%, suggesting that while the model could still learn from the reduced data, it lost crucial features that contributed to classification accuracy. However, the loss metric improved from 1.0705 to 0.5905, implying that the PCA-applied model was more stable but may have lost significant details essential for accurate classification.

A trade-off was observed between dataset size and model performance. The primary advantage of dimensionality reduction was the reduction in computational complexity, making the model more efficient in terms of processing power and memory usage. However, this efficiency came at the cost of classification accuracy.

Final conclusions drawn from this analysis indicate that if the primary objective is computational efficiency and reduced memory requirements, PCA can be a viable approach. However, careful selection of the optimal number of principal components is necessary to minimize information loss. On the other hand, if achieving maximum accuracy is the priority, it is preferable to use the original dataset without dimensionality reduction or explore alternative techniques such as feature selection instead of PCA.

Discussion and Conclusions

The results of this study demonstrate that Convolutional Neural Networks outperform traditional machine learning models in image classification tasks involving cats and dogs The CNN model achieved an accuracy of 73% compared to 60% obtained by the strongest classical model Random Forest and lower performance by various SVM kernels The ability of CNNs to learn hierarchical representations directly from images without the need for handcrafted features gives them a clear advantage over classical methods This study confirms that deep learning approaches particularly CNNs are well-suited for complex image classification tasks

One key observation from this study is the significant impact of hyperparameter tuning on CNN performance The systematic exploration of parameters such as the number of epochs dropout rate and dense layer units was instrumental in improving accuracy from the initial model configuration to its final optimized state The best-performing model was configured with 30 epochs 256 dense units and a dropout rate of 0.5 which led to the highest observed accuracy of 77% This highlights the importance of fine-tuning hyperparameters in deep learning models to achieve optimal performance

Dataset modifications such as data augmentation and imbalanced class distributions also played a critical role in model performance The introduction of additional augmented images did not necessarily improve classification accuracy Instead it led to an increase in model bias particularly favoring one class over another This suggests that while augmentation can be a powerful tool in increasing dataset diversity careful tuning of augmentation parameters is necessary to prevent unintended biases The dataset imbalance experiments further demonstrated that models trained on highly imbalanced datasets tend to achieve higher accuracy but at the cost of reduced generalization Models trained on a balanced dataset exhibited better differentiation capabilities but had lower raw accuracy reinforcing the trade-off between generalization and overfitting to the dominant class

Additionally architectural modifications were explored to stabilize training and improve performance Key changes such as implementing batch normalization and replacing the Flatten layer with GlobalAveragePooling2D contributed to reduced overfitting and improved generalization The introduction of explicit input layers and adjustments to dropout rates also contributed to better convergence and model robustness The experimental results confirm that these architectural refinements were essential in ensuring model stability and efficiency

Despite the promising results obtained in this study there are several limitations that warrant further exploration The dataset size though adequate could be expanded to include more diverse images to improve model generalization The study primarily focused on a binary classification task but future work could explore multi-class classification involving additional animal categories Furthermore while CNN-based models performed exceptionally well alternative deep learning architectures such as Vision Transformers or more advanced CNN variants like EfficientNet and ResNeXt could be explored for further accuracy improvements

Future Work

Future work should focus on expanding the dataset to include a larger and more diverse set of images which can help improve model robustness and generalization Additional experiments should be conducted using transfer learning techniques by leveraging pre-trained models such as ResNet DenseNet and EfficientNet to assess their impact on classification accuracy compared to training a CNN from scratch These models have been shown to perform well on large-scale image datasets and could further enhance the performance of cat and dog classification tasks

Another avenue for future research involves experimenting with different data augmentation techniques beyond basic transformations such as applying generative adversarial networks GANs to generate synthetic but realistic images to enhance training diversity Investigating various class balancing strategies such as using focal loss or adjusting class weights during training could help mitigate bias observed in the imbalance experiments

Architectural modifications should also be explored further by increasing network depth incorporating attention mechanisms and testing different pooling strategies These refinements could improve feature extraction and classification performance Testing different optimization algorithms such as AdamW or Ranger may also enhance training stability and convergence speed leading to a more robust model

Lastly deploying the trained CNN model in a real-world application such as a mobile or web-based classification tool could provide valuable insights into its practical usability Model deployment would also enable testing on real-world images with various environmental factors ensuring that the network maintains high accuracy under different conditions This step would move the model from a purely academic exercise to a functional system capable of practical implementation

References

Boureau Y L Ponce J & LeCun Y 2010 A theoretical analysis of feature pooling in visual recognition Proceedings of the International Conference on Machine Learning ICML

Goodfellow I Bengio Y & Courville A 2016 Deep Learning MIT Press

Hanley J A & McNeil B J 1982 The meaning and use of the area under a receiver operating characteristic ROC curve Radiology

He K Zhang X Ren S & Sun J 2016 Deep residual learning for image recognition Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR

Krizhevsky A Sutskever I & Hinton G 2012 ImageNet classification with deep convolutional neural networks Advances in neural information processing systems NeurIPS

LeCun Y Bottou L Bengio Y & Haffner P 1998 Gradient-based learning applied to document recognition Proceedings of the IEEE

Nair V & Hinton G E 2010 Rectified linear units improve restricted Boltzmann machines Proceedings of the International Conference on Machine Learning ICML

Simonyan K & Zisserman A 2015 Very deep convolutional networks for large-scale image recognition International Conference on Learning Representations ICLR

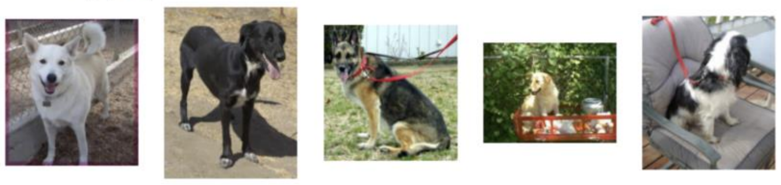
Srivastava N Hinton G Krizhevsky A Sutskever I & Salakhutdinov R 2014 Dropout A simple way to prevent neural networks from overfitting Journal of Machine Learning Research 15 1 1929-1958

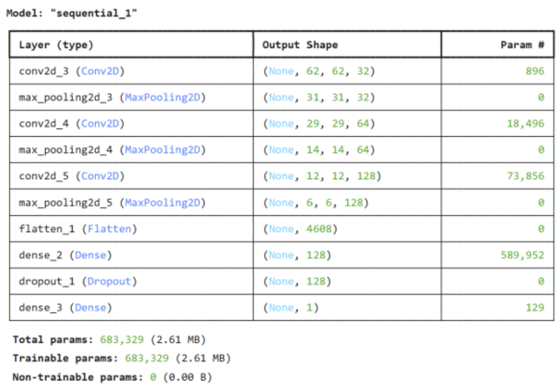
Tan M & Le Q 2019 EfficientNet Rethinking model scaling for convolutional neural networks Proceedings of the International Conference on Machine Learning ICML

Appendices

דוגמאות של cats (Test):

דוגמאות של dogs (Test):





Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_4 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_5 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_2 (Dense)	(None, 128)	589,952
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 683,329 (2.61 MB)
Trainable params: 683,329 (2.61 MB)
Non-trainable params: 0 (0.00 B)

- ביצועים עבור CNN:

```
              precision    recall  f1-score   support

        Cats       0.72      0.76      0.74       149
        Dogs       0.75      0.70      0.72       151

    accuracy                           0.73       300
   macro avg       0.73      0.73      0.73       300
weighted avg       0.73      0.73      0.73       300

 Confusion Matrix:
 [[113  36]
 [ 45 106]]
```

- ביצועים עבור SVM (Linear Kernel):

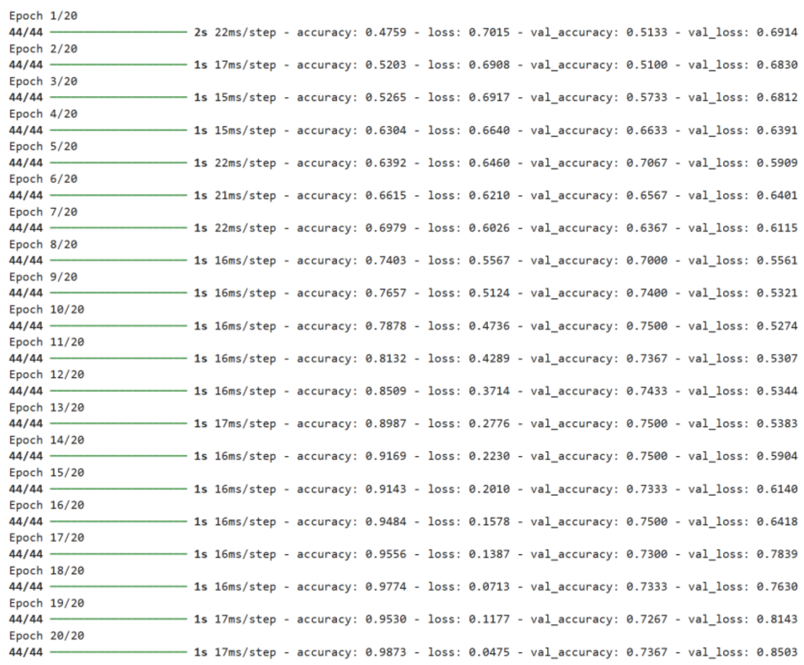
```
              precision    recall  f1-score   support

        Cats       0.50      0.56      0.53       149
        Dogs       0.50      0.44      0.47       151

    accuracy                           0.50       300
   macro avg       0.50      0.50      0.50       300
weighted avg       0.50      0.50      0.50       300

 Confusion Matrix:
 [[83 66]
 [84 67]]
```

- ביצועים עבור SVM (RBF Kernel):

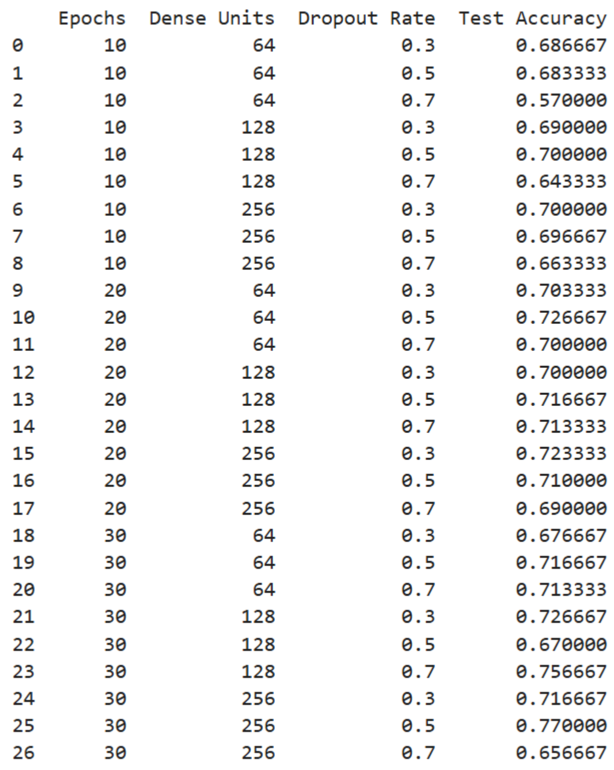
```
              precision    recall  f1-score   support

        Cats       0.59      0.56      0.57       149
        Dogs       0.59      0.62      0.60       151

    accuracy                           0.59       300
   macro avg       0.59      0.59      0.59       300
weighted avg       0.59      0.59      0.59       300

 Confusion Matrix:
 [[83 66]
 [57 94]]
```

- ביצועים עבור SVM (Polynomial Kernel):

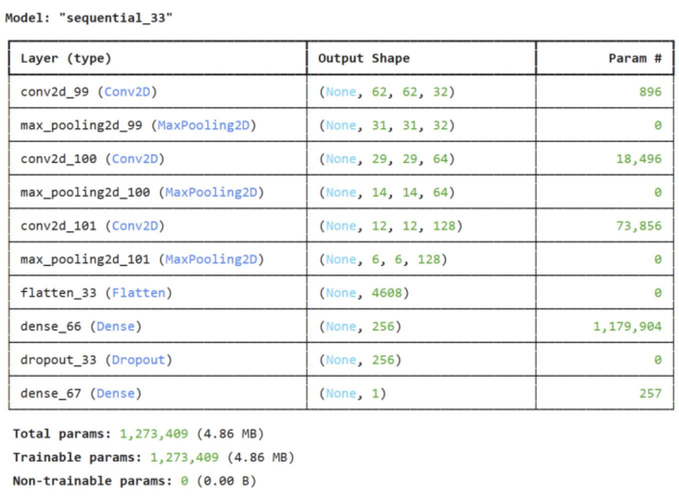
```
              precision    recall  f1-score   support

        Cats       0.54      0.58      0.56       149
        Dogs       0.55      0.51      0.53       151

    accuracy                           0.55       300
   macro avg       0.55      0.55      0.55       300
weighted avg       0.55      0.55      0.55       300

 Confusion Matrix:
 [[87 62]
 [74 77]]
```



```
Epoch 1/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - accuracy: 0.4759 - loss: 0.7015 - val_accuracy: 0.5133 - val_loss: 0.6914
Epoch 2/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.5203 - loss: 0.6908 - val_accuracy: 0.5100 - val_loss: 0.6830
Epoch 3/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.5265 - loss: 0.6917 - val_accuracy: 0.5733 - val_loss: 0.6812
Epoch 4/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - accuracy: 0.6304 - loss: 0.6640 - val_accuracy: 0.6633 - val_loss: 0.6391
Epoch 5/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6392 - loss: 0.6460 - val_accuracy: 0.7067 - val_loss: 0.5909
Epoch 6/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 21ms/step - accuracy: 0.6615 - loss: 0.6210 - val_accuracy: 0.6567 - val_loss: 0.6401
Epoch 7/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6979 - loss: 0.6026 - val_accuracy: 0.6367 - val_loss: 0.6115
Epoch 8/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7403 - loss: 0.5567 - val_accuracy: 0.7000 - val_loss: 0.5561
Epoch 9/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7657 - loss: 0.5124 - val_accuracy: 0.7400 - val_loss: 0.5321
Epoch 10/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.7878 - loss: 0.4736 - val_accuracy: 0.7500 - val_loss: 0.5274
Epoch 11/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8132 - loss: 0.4289 - val_accuracy: 0.7367 - val_loss: 0.5307
Epoch 12/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.8509 - loss: 0.3714 - val_accuracy: 0.7433 - val_loss: 0.5344
Epoch 13/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.8987 - loss: 0.2776 - val_accuracy: 0.7500 - val_loss: 0.5383
Epoch 14/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9169 - loss: 0.2230 - val_accuracy: 0.7500 - val_loss: 0.5904
Epoch 15/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9143 - loss: 0.2010 - val_accuracy: 0.7333 - val_loss: 0.6140
Epoch 16/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9484 - loss: 0.1578 - val_accuracy: 0.7500 - val_loss: 0.6418
Epoch 17/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9556 - loss: 0.1387 - val_accuracy: 0.7300 - val_loss: 0.7839
Epoch 18/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - accuracy: 0.9774 - loss: 0.0713 - val_accuracy: 0.7333 - val_loss: 0.7630
Epoch 19/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.9530 - loss: 0.1177 - val_accuracy: 0.7267 - val_loss: 0.8143
Epoch 20/20
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - accuracy: 0.9873 - loss: 0.0475 - val_accuracy: 0.7367 - val_loss: 0.8503
```



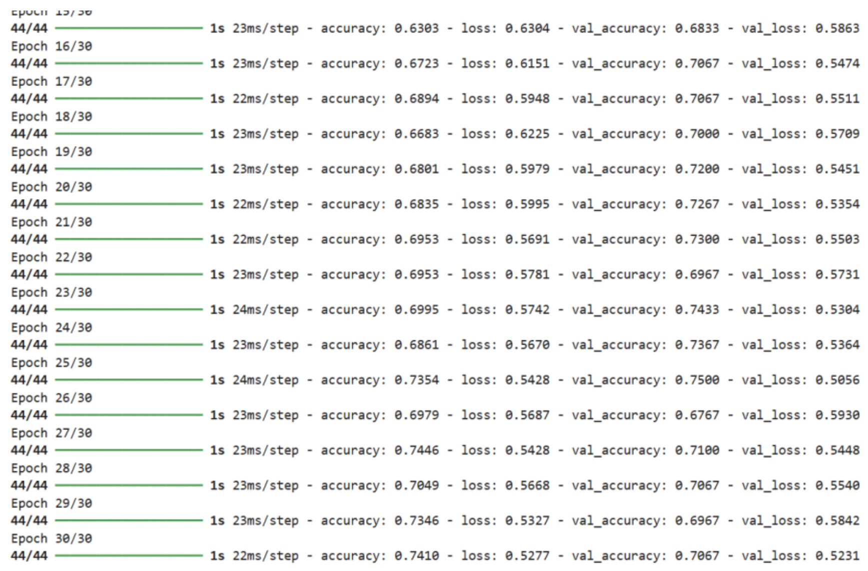
```
    Epochs  Dense Units  Dropout Rate  Test Accuracy
0       10           64           0.3       0.686667
1       10           64           0.5       0.683333
2       10           64           0.7       0.570000
3       10          128           0.3       0.690000
4       10          128           0.5       0.700000
5       10          128           0.7       0.643333
6       10          256           0.3       0.700000
7       10          256           0.5       0.696667
8       10          256           0.7       0.663333
9       20           64           0.3       0.703333
10      20           64           0.5       0.726667
11      20           64           0.7       0.700000
12      20          128           0.3       0.700000
13      20          128           0.5       0.716667
14      20          128           0.7       0.713333
15      20          256           0.3       0.723333
16      20          256           0.5       0.710000
17      20          256           0.7       0.690000
18      30           64           0.3       0.676667
19      30           64           0.5       0.716667
20      30           64           0.7       0.713333
21      30          128           0.3       0.726667
22      30          128           0.5       0.670000
23      30          128           0.7       0.756667
24      30          256           0.3       0.716667
25      30          256           0.5       0.770000
26      30          256           0.7       0.656667
```



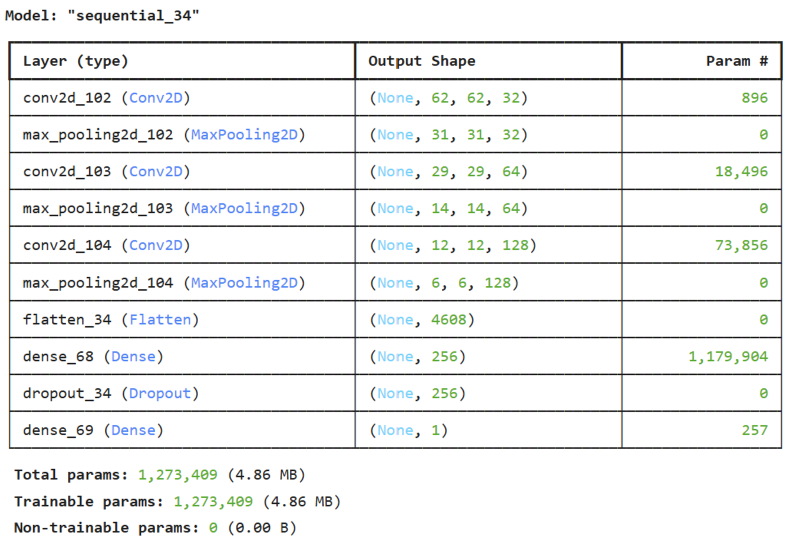
Model: "sequential_33"

Layer (type)	Output Shape	Param #
conv2d_99 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_99 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_100 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_100 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_101 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_101 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten_33 (Flatten)	(None, 4608)	0
dense_66 (Dense)	(None, 256)	1,179,904
dropout_33 (Dropout)	(None, 256)	0
dense_67 (Dense)	(None, 1)	257

Total params: 1,273,409 (4.86 MB)
Trainable params: 1,273,409 (4.86 MB)
Non-trainable params: 0 (0.00 B)



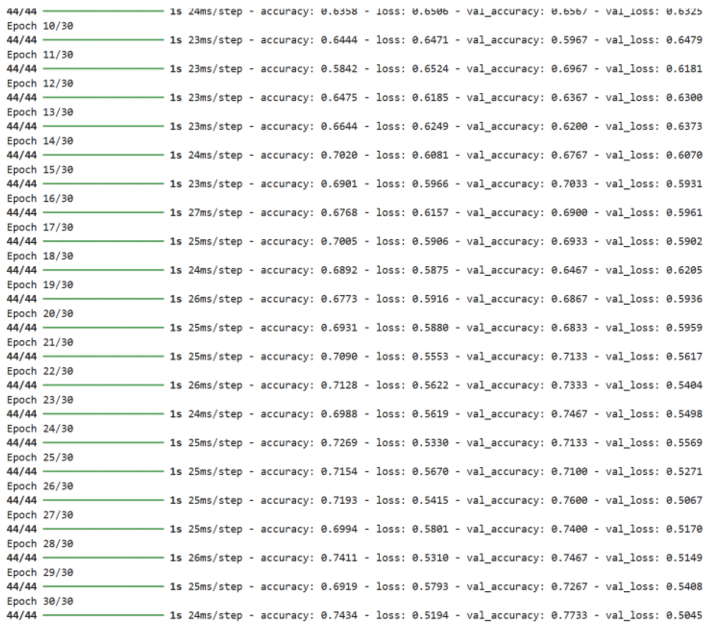
```
Epoch 15/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6303 - loss: 0.6304 - val_accuracy: 0.6833 - val_loss: 0.5863
Epoch 16/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6723 - loss: 0.6151 - val_accuracy: 0.7067 - val_loss: 0.5474
Epoch 17/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6894 - loss: 0.5948 - val_accuracy: 0.7067 - val_loss: 0.5511
Epoch 18/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6683 - loss: 0.6225 - val_accuracy: 0.7000 - val_loss: 0.5709
Epoch 19/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6801 - loss: 0.5979 - val_accuracy: 0.7200 - val_loss: 0.5451
Epoch 20/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6835 - loss: 0.5995 - val_accuracy: 0.7267 - val_loss: 0.5354
Epoch 21/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.6953 - loss: 0.5691 - val_accuracy: 0.7300 - val_loss: 0.5503
Epoch 22/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6953 - loss: 0.5781 - val_accuracy: 0.6967 - val_loss: 0.5731
Epoch 23/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.6995 - loss: 0.5742 - val_accuracy: 0.7433 - val_loss: 0.5304
Epoch 24/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6861 - loss: 0.5670 - val_accuracy: 0.7367 - val_loss: 0.5364
Epoch 25/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.7354 - loss: 0.5428 - val_accuracy: 0.7500 - val_loss: 0.5056
Epoch 26/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6979 - loss: 0.5687 - val_accuracy: 0.6767 - val_loss: 0.5930
Epoch 27/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.7446 - loss: 0.5428 - val_accuracy: 0.7100 - val_loss: 0.5448
Epoch 28/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.7049 - loss: 0.5668 - val_accuracy: 0.7067 - val_loss: 0.5540
Epoch 29/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.7346 - loss: 0.5327 - val_accuracy: 0.6967 - val_loss: 0.5842
Epoch 30/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 22ms/step - accuracy: 0.7410 - loss: 0.5277 - val_accuracy: 0.7067 - val_loss: 0.5231
```



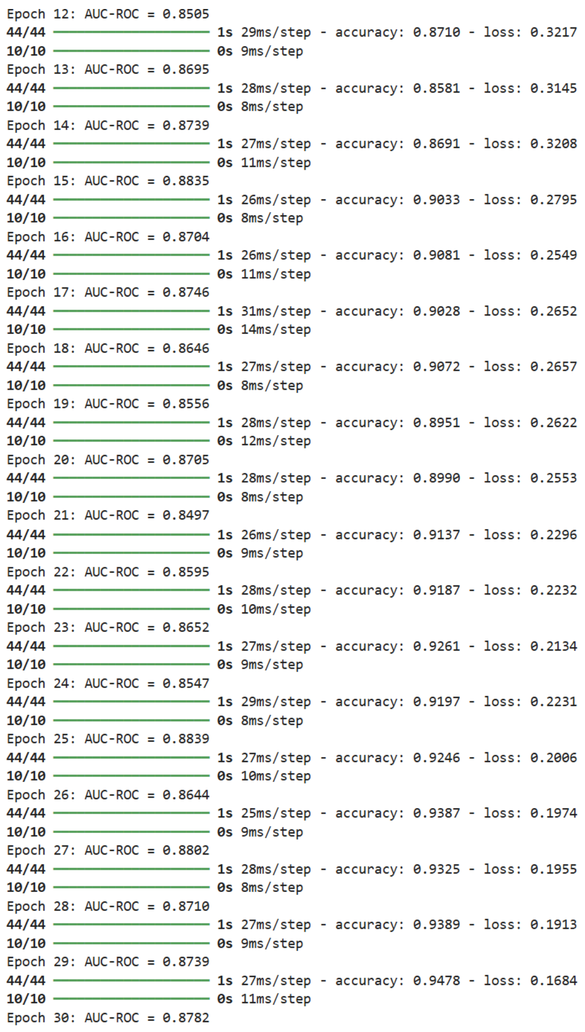
Model: "sequential_34"

Layer (type)	Output Shape	Param #
conv2d_102 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_102 (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_103 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_103 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_104 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_104 (MaxPooling2D)	(None, 6, 6, 128)	0
flatten_34 (Flatten)	(None, 4608)	0
dense_68 (Dense)	(None, 256)	1,179,904
dropout_34 (Dropout)	(None, 256)	0
dense_69 (Dense)	(None, 1)	257

Total params: 1,273,409 (4.86 MB)
Trainable params: 1,273,409 (4.86 MB)
Non-trainable params: 0 (0.00 B)



```
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.6358 - loss: 0.6506 - val_accuracy: 0.6567 - val_loss: 0.6325
Epoch 10/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6444 - loss: 0.6471 - val_accuracy: 0.5967 - val_loss: 0.6479
Epoch 11/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.5842 - loss: 0.6524 - val_accuracy: 0.6967 - val_loss: 0.6181
Epoch 12/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6475 - loss: 0.6185 - val_accuracy: 0.6367 - val_loss: 0.6300
Epoch 13/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6644 - loss: 0.6249 - val_accuracy: 0.6200 - val_loss: 0.6373
Epoch 14/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.7020 - loss: 0.6081 - val_accuracy: 0.6767 - val_loss: 0.6070
Epoch 15/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 23ms/step - accuracy: 0.6901 - loss: 0.5966 - val_accuracy: 0.7033 - val_loss: 0.5931
Epoch 16/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.6768 - loss: 0.6157 - val_accuracy: 0.6900 - val_loss: 0.5961
Epoch 17/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.7005 - loss: 0.5906 - val_accuracy: 0.6933 - val_loss: 0.5902
Epoch 18/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.6892 - loss: 0.5875 - val_accuracy: 0.6467 - val_loss: 0.6205
Epoch 19/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.6773 - loss: 0.5916 - val_accuracy: 0.6867 - val_loss: 0.5936
Epoch 20/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.6931 - loss: 0.5880 - val_accuracy: 0.6833 - val_loss: 0.5959
Epoch 21/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.7090 - loss: 0.5553 - val_accuracy: 0.7133 - val_loss: 0.5617
Epoch 22/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.7128 - loss: 0.5622 - val_accuracy: 0.7333 - val_loss: 0.5404
Epoch 23/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.6988 - loss: 0.5619 - val_accuracy: 0.7467 - val_loss: 0.5498
Epoch 24/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.7269 - loss: 0.5330 - val_accuracy: 0.7133 - val_loss: 0.5569
Epoch 25/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.7154 - loss: 0.5670 - val_accuracy: 0.7100 - val_loss: 0.5271
Epoch 26/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.7193 - loss: 0.5415 - val_accuracy: 0.7600 - val_loss: 0.5067
Epoch 27/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.6994 - loss: 0.5801 - val_accuracy: 0.7400 - val_loss: 0.5170
Epoch 28/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.7411 - loss: 0.5310 - val_accuracy: 0.7467 - val_loss: 0.5149
Epoch 29/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.6919 - loss: 0.5793 - val_accuracy: 0.7267 - val_loss: 0.5408
Epoch 30/30
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - accuracy: 0.7434 - loss: 0.5194 - val_accuracy: 0.7733 - val_loss: 0.5045
```



```
Epoch 12: AUC-ROC = 0.8505
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.8710 - loss: 0.3217
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 13: AUC-ROC = 0.8695
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8581 - loss: 0.3145
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 14: AUC-ROC = 0.8739
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8691 - loss: 0.3208
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
Epoch 15: AUC-ROC = 0.8835
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9033 - loss: 0.2795
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 16: AUC-ROC = 0.8704
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9081 - loss: 0.2549
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
Epoch 17: AUC-ROC = 0.8746
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.9028 - loss: 0.2652
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
Epoch 18: AUC-ROC = 0.8646
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9072 - loss: 0.2657
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 19: AUC-ROC = 0.8556
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8951 - loss: 0.2622
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
Epoch 20: AUC-ROC = 0.8705
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8990 - loss: 0.2553
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 21: AUC-ROC = 0.8497
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9137 - loss: 0.2296
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 22: AUC-ROC = 0.8595
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9187 - loss: 0.2232
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Epoch 23: AUC-ROC = 0.8652
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9261 - loss: 0.2134
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 24: AUC-ROC = 0.8547
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.9197 - loss: 0.2231
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 25: AUC-ROC = 0.8839
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9246 - loss: 0.2006
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Epoch 26: AUC-ROC = 0.8644
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - accuracy: 0.9387 - loss: 0.1974
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 27: AUC-ROC = 0.8802
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9325 - loss: 0.1955
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
Epoch 28: AUC-ROC = 0.8710
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9389 - loss: 0.1913
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step
Epoch 29: AUC-ROC = 0.8739
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.9478 - loss: 0.1684
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step
Epoch 30: AUC-ROC = 0.8782
```