Name: Tien Duong, Kumuda Valli

Cat and Dog Image Classification Using Convolutional Neural Network

Abstract:

In this project, we designed and implemented an Artificial Neural Network (ANN) to perform binary image classification, distinguishing between images of dogs and cats. We adopted a Convolutional Neural Network (CNN) architecture and a DenseNet architecture. While CNNs are widely recognized for their effectiveness in image recognition tasks due to their ability to automatically learn spatial hierarchies of features, the DenseNet architecture was chosen to propagate features and gradients through dense connectivity efficiently. Despite working with a relatively small dataset and limiting the number of training epochs, our model demonstrated promising performance. Through careful model tuning and optimization, we achieved a classification accuracy of 71%.

Furthermore, the model yielded a precision of 0.78, a recall of 0.64, and an F1-score of 0.70, indicating a balanced trade-off between false positives and false negatives. The Area Under the Receiver Operating Characteristic Curve (AUC-ROC) was measured at 0.81, reflecting a strong overall capability in distinguishing between the two classes. These results underscore the potential of even relatively simple deep learning models to effectively tackle image classification problems when built with a sound architectural approach.

Introduction:

Convolutional Neural Networks (CNNs) are a popular and robust approach for tackling image classification problems because they automatically and efficiently learn spatial hierarchies of features from raw pixel data. Using convolutional layers to detect edges, textures, and patterns at multiple scales, CNNs can capture essential visual information contributing to high classification accuracy. This makes them particularly effective for tasks such as object detection, facial recognition, medical imaging, and, in our case, distinguishing between images of dogs and cats.

DenseNet (Densely Connected Convolution Networks) is an extension of CNNs designed to enhance feature propagation throughout the network. This model introduces dense connectivity, where each layer receives inputs from all preceding layers and solves the vanishing gradient problem. Thus, DenseNet can achieve higher accuracy with fewer parameters than CNNs.

The binary classification of cats and dogs is a classic benchmark problem in computer vision and deep learning. Although simple in concept, it poses meaningful challenges due to the wide variation in animal poses, lighting conditions, backgrounds, and image quality. This task serves as a valuable entry point into CNN-based image classification, enabling practitioners to explore key aspects of deep learning such as model architecture design, data preprocessing, training optimization, and evaluation metrics.

In this project, we designed and implemented a CNN model using **PyTorch**, an open-source deep learning framework known for its flexibility and ease of use. PyTorch

enabled us to efficiently construct, train, and evaluate our model, while providing fine-grained control over the architecture and training loop. Given practical limitations in terms of hardware and storage, we resized all images to 32×32 pixels and trained a relatively lightweight model. Despite these constraints, our model achieved strong performance by leveraging CNN's strength in capturing low- to mid-level visual features.

Dataset:

We utilized a dataset containing over 32,000 labeled images of cats and dogs, sourced from a publicly available Kaggle competition. To manage storage constraints and ensure efficient computation, all photos were resized to 32×32 pixels. Although this resolution is relatively low, it preserves essential visual features such as edges, contours, and basic textures, sufficient for a Convolutional Neural Network (CNN) or a DenseNet to learn meaningful patterns for classification. Images are 3-RGB channel, as we want to capture actual cat/dog images instead of using grayscale.

Before feeding the images into the model, we applied several preprocessing steps to improve learning efficiency and model performance. All images were normalized by scaling pixel values to the range [0, 1], which helps stabilize and speed up the training process. The dataset was then shuffled to prevent the model from learning unintended patterns based on image order. Finally, we split the dataset into training and testing sets—typically using an 80/20 ratio—to ensure the model could be evaluated on unseen data and to assess its generalization ability.

Training process:

We trained two models for this project to compare which architecture best suits this dataset. The first is a traditional CNN model, whereas the second is a DenseNet architecture. Both models optimize weights by backpropagation. We explored multiple optimization algorithms like Stochastic Gradient Descent, RMSprop, Adam (adaptive moment estimation), and AdamW, and finally decided on RMSprop. The following metrics resulted when we tried using various other optimizing algorithms:

Optimization algorithm	**Accuracy(%)**
RMSprop	71.2%
Adam	70%
AdamW	70%
SGD	68%

While training the models, we faced overfitting issues when we increased the number of epochs to 20. We achieved high training accuracy, but the model could not generalize to the testing dataset.

In the CNN model, we implemented cross-validation to evaluate the model over different subsets of data. Based on this, we set up an early stopping condition, effectively preventing overfitting and decreasing validation loss. When we applied the same logic to our DenseNet model, we noticed that the early stopping condition was triggered after 8 epochs. However, we noticed that we got better metrics after 10 epochs. So we continued training until 10 epochs.

Furthermore, we applied data augmentation techniques like horizontal flips and rotations. Dropout layers were also incorporated in both models.

Evaluation metrics and visualization:

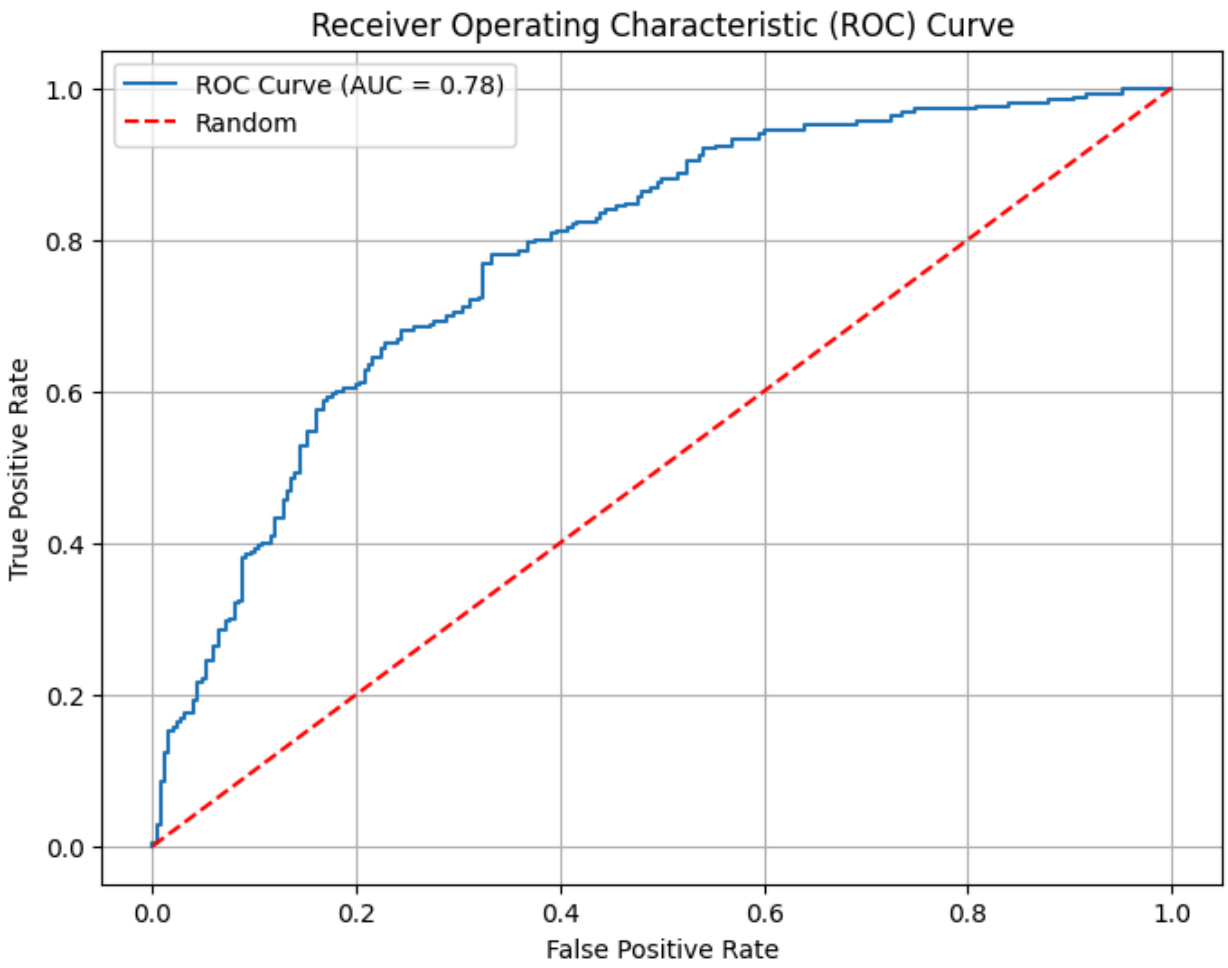
Both models have been evaluated on the following metrics: accuracy, precision, recall, F1 score, and AUC (Area Under Curve). They're all crucial in assessing the model's performance. Accuracy often doesn't give us the whole picture. Precision and recall usually step in in such situations and provide a deeper insight. The F1 score focuses on the balance between these two.

Evaluation metrics:

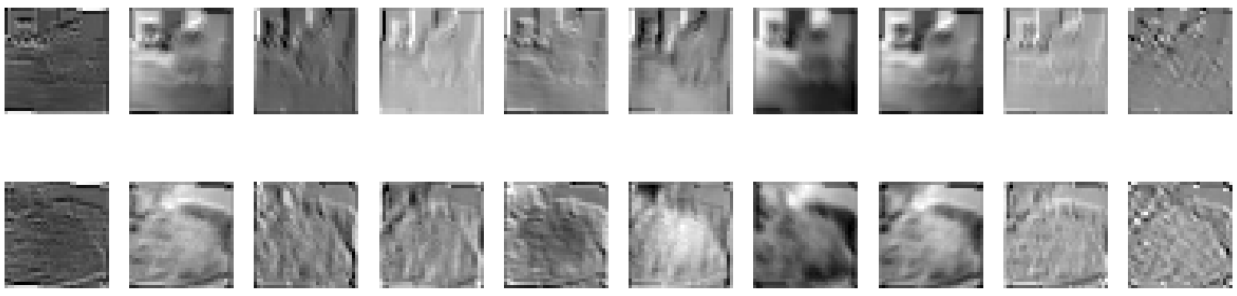
Metric	**CNN**	**DenseNet**
Accuracy	71.20%	76.6%
Precision	0.68	0.79
Recall	0.80	0.73
F1 score	0.74	0.76
AUC	0.78	0.84

The DenseNet had better precision and recall due to its advanced architecture. Below, we have included some visualizations, like ROC graphs and feature maps, for a better picture.

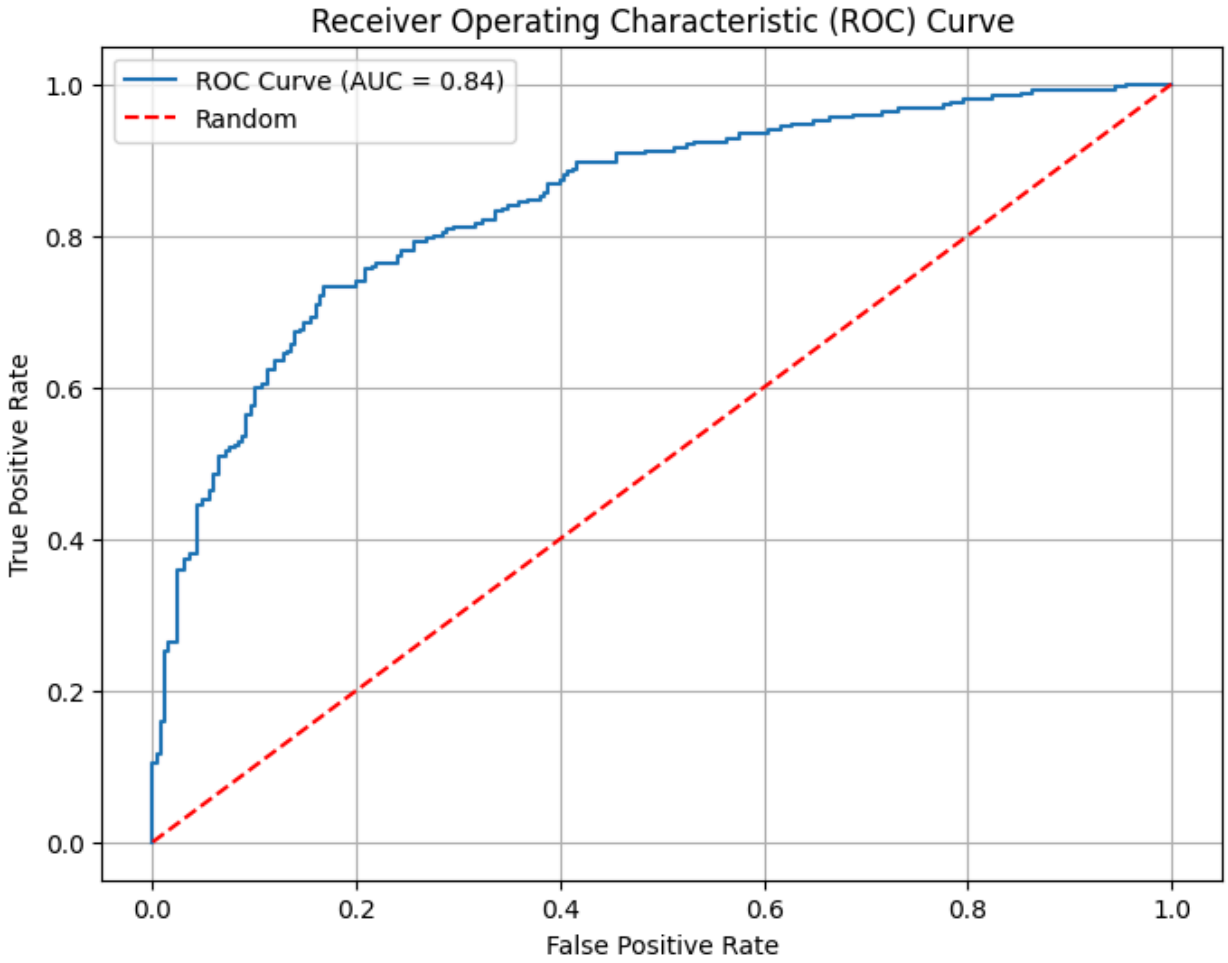
ROC curve for the CNN model:



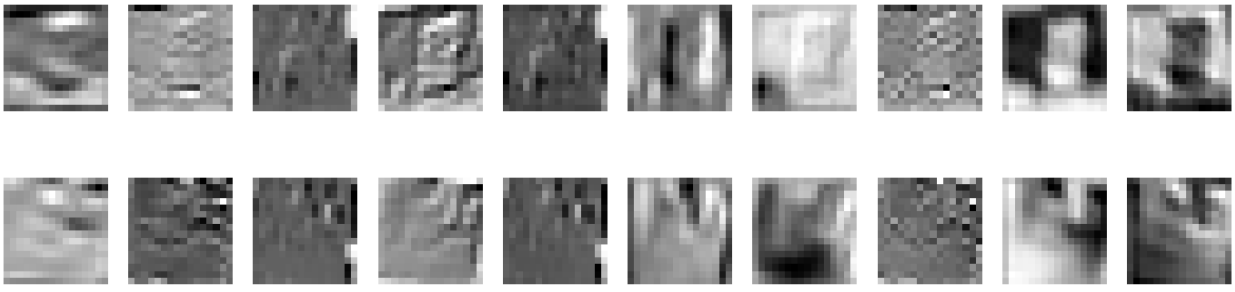
Feature maps for the CNN model:



ROC curve for the DenseNet model:



Feature maps for the DenseNet model:



Improvement:

Several constraints limited the full potential of our model's performance. Due to time constraints, we could only train the model for 20 epochs and used a subset of the available dataset for training. These factors directly impacted the model's generalizability, leading to suboptimal accuracy. With more training time and the ability to fully utilize the entire dataset, we anticipate a notable improvement in model performance and generalization capability.

In addition, all images were resized to a resolution of 32×32 pixels to manage storage and memory usage. While this made the model more computationally efficient, it also limited the detail and richness of the features the model could learn. Given access to greater hardware resources, using higher-resolution images would allow the model to capture more fine-grained visual information, which could lead to improved classification accuracy.

Another potential improvement is the use of **data augmentation** techniques during training. Applying transformations such as horizontal flipping, random cropping, rotation, zooming, and brightness adjustment can effectively expand the training dataset and introduce greater variability. This helps prevent overfitting and improves the model's robustness to real-world variations in input data. Incorporating data augmentation would be a low-cost but highly effective strategy to enhance model performance without requiring additional labeled data.

Conclusion:

In this project, we demonstrated using a Convolutional Neural Network (CNN) to classify images of dogs and cats. Despite working with limited training time, a reduced dataset, and low-resolution images, our model achieved promising results across multiple performance metrics, including accuracy, precision, recall, and AUC. These outcomes highlight the effectiveness of CNNs even under constrained conditions.

While there is still significant room for improvement, such as training for more epochs, utilizing the whole dataset, employing higher-resolution images, and incorporating data augmentation, the results achieved thus far validate the potential of our approach. With these enhancements, we expect the model to perform even better, making it a solid foundation for future development and experimentation in image classification tasks.

Looking ahead, we plan to explore more advanced architectures, such as deeper CNNs or architectures like ResNet or VGG, which can extract more complex features and improve performance further. Additionally, **transfer learning** presents an exciting opportunity for applying pre-trained models to new classification tasks. For instance, a model like **VGG16** or **InceptionV3**, which has been pre-trained on large datasets like ImageNet, can be fine-tuned for animal classification tasks, such as identifying various species of animals. By leveraging the feature extraction capabilities of these pre-trained models, we can significantly reduce training time while improving accuracy, even when working with limited labeled data.