Advanced Machine Learning

Assignment 2 – Convolution Networks

Krishna Kumar Tavva

Cats Vs Dogs:

# Objective:

To build a convolutional neural network (CNN) that can recognize cat and dog images and to understand the impact of training size during the model-building phase.

# Model Building:

In total, we constructed 15 models with various layers, nodes, optimizers, and other hyperparameters.

We categorized these models into two groups: **Scratch Models** and **Pre-Trained Models**. Below, we provide the hyperparameter configurations and performance evaluations of the models trained from scratch, along with the findings from these models.

# Hyper Tunning Parameters: (Scratch Models)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| #No. | Input  Layers | Filters | Filter Size | Optimizer | Training | Validation  & Test | Dropout |
| Model 1 | 5 | 32 to 256 | 3 | Adam | 1000 | 500 & 500 | - |
| Model 2 | 5 | 32 to 256 | 3 | Adam | 1000 | 500 & 500 | 0.3 |
| Model 3 | 6 | 32 to 512 | 3 | Adam | 1000 | 500 & 500 | 0.3 |
| Model 4 | 5 | 64 to 1024 | 3 | Adam | 1000 | 500 & 500 | 0.5 |
| Model 5 | 5 | 32 to 256 | 3 | Adam | 1500 | 500 & 500 | 0.3 |
| Model 6 | 5 | 32 to 256 | 3 | Adam | 1500 | 500 & 500 | 0.3 |
| Model 7 | 5 | 32 to 256 | 3 | Adam | 2000 | 500 & 500 | 0.3 |
| Model 8 | 5 | 32 to 256 | 3 | Adam | 2000 | 500 & 500 | 0.3 |
| Model 9 | 5 | 32 to 512 | 3 | Adam | 2000 | 500 & 500 | 0.3 |
| **Model 10** | **5** | **32 to 256** | **3** | **Adam** | **3000** | **500 & 500** | **0.3** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #No. | Max Pooling | Strides | Padding | Augmented Images | Test Performance  (Loss, Accuracy) |
| Model 1 | Yes (Pool Size = 2) | - | - | - | (0.676, 0.598) |
| Model 2 | Yes (Pool Size = 2) | - | - | Yes | (0.567, 0.720) |
| Model 3 | Yes (Pool Size = 2) | - | - | Yes | (0.616, 0.674) |
| Model 4 | Yes (Pool Size = 2) | - | - | Yes | (0.629, 0.648) |
| Model 5 | Yes (Pool Size = 2) | - | - | Yes | (0.478, 0.764) |
| Model 6 | - | Yes (Strides = 2) | - | Yes | (0.667, 0.610) |
| Model 7 | Yes (Pool Size = 2) | - | - | Yes | (0.476, 0.765) |
| Model 8 | Yes (Pool Size = 2) | Yes (Strides = 2) | - | Yes | (0.454, 0.779) |
| Model 9 | Yes (Pool Size = 2) | Yes (Strides = 2) | Yes (Same) | Yes | (0.693, 0.500) |
| **Model 10** | **Yes (Pool Size = 2)** | **-** | **-** | **Yes** | **(0.489, 0.750)** |

**Findings:**

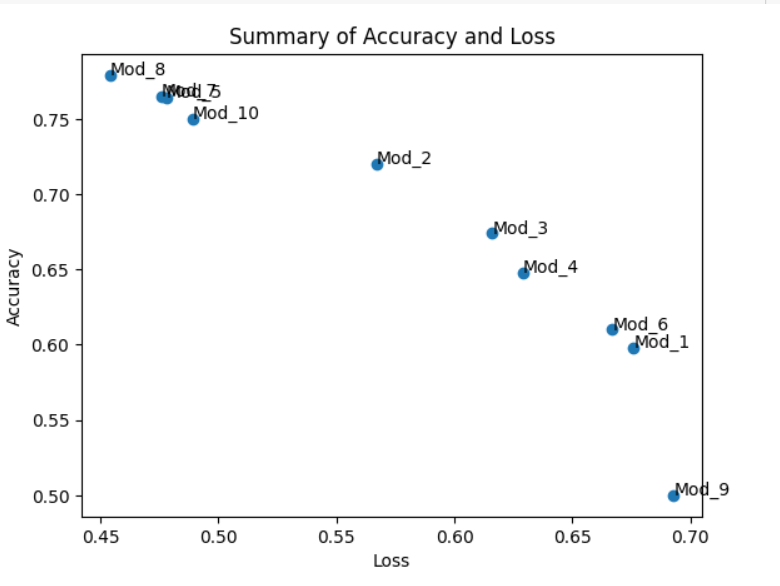
* When the training size was 1000 samples, the first four models did not achieve high accuracy. However, Model 2, which was trained with augmented images, showed the best accuracy among these models. This indicates that data augmentation is an effective technique for improving the model's performance.
* Models 3 and 4 had changes to their filters and layers. Model 3 consisted of 6 layers with filters ranging from 32, 64, 128, 256, 256, and 512. Model 4 had 5 layers with filters ranging from 64, 128, 256, 512, and 1024. However, increasing the number of filters and adding more layers did not result in improved model performance.
* The training sample size was further increased to 2000 samples for Models 5 and 6. Here, Model 5 achieved 76.4% accuracy and 47.8% loss. This was the same model that initially received 72.0% accuracy with 1000 training samples.

As the model was exposed to more samples, along with the augmented versions of the images, it was able to learn more effectively, thereby improving its ability to correctly recognize the images.

* Model 6 didn’t have a pooling layer instead of it we had strides as a mechanism to decrease the spatial dimensionality. This change in mechanism from pooling to strides alone didn’t show any significant improvement in the model’s performance.
* Model 7 was a replica of the previous Models 2 and 5, with the only difference being an increase in the training sample size to 2000. Interestingly, when the training sample size was increased from 1500 to 2000, the model's accuracy decreased slightly from 76.4% to 76.5%. This suggests that merely increasing the sample size does not always improve performance. It highlights the importance of choosing the right sample size to ensure the model trains effectively and generalizes well on unseen data.
* Model 8 was built using a combination of Max Pooling and Strides, which proved to be an effective approach. This combination led to an increase in accuracy, reaching 77.9%.
* Model 9, which included padding, did not perform well, showing a decline in accuracy.
* Finally, Model 10 achieved an improved accuracy of 75% when the sample size was increased to 3000.

# Final Comments:

# To answer the main question regarding the relationship between training sample size and the choice of network, we can conclude that there is a significant connection between these two factors. The size of the training sample and the architecture of the network both play crucial roles in the performance of the model. Using Max Pooling operations, incorporating Dropout, and employing augmented images proved to be efficient strategies across multiple models, leading to notable improvements in accuracy and overall model performance.



# Pre-Trained Network:

For the image recognition task, we utilized the VGG-16 as a pre-trained network. VGG-16 is a robust and widely-used model trained on millions of images across various categories, making it a powerful tool for image recognition tasks. In this project, we built five models, two of which were hyper-tuned versions of the initial models.

# Hyper Tunning Parameters: (Pre-Trained Models)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| #No. | Dense Layer | Training  Size | Optimizer | Validation  & Test | Dropout |
| Model 1 | 1 (256 Nodes) | 1000 | rmsprop | 500 & 500 | 0.3 |
| Model 2 | 1 (256 Nodes) | 1000 | rmsprop | 500 & 500 | 0.3 |
| Model 3 | 1 (256 Nodes) | 1000 | rmsprop | 500 & 500 | 0.3 |
| Model 4 | 1 (256 Nodes) | 3000 | Adam | 500 & 500 | 0.3 |
| Model 5 | 1 (256 Nodes) | 3000 | Adam (1e-5) | 500 & 500 | 0.3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| #No. | Pre-Trained Weights Update  Set to False | Freeze Layers | Augmented Images | Test Performance  (Loss, Accuracy) |
| Model 1 | No | False | No | (4.206, 0.978) |
| Model 2 | Yes | False | Yes | (0.708, 0.536) |
| Model 3 | Yes | True | Yes | (0.791, 0.390) |
| Model 4 | Yes | False | Yes | (0.178, 0.974) |
| Model 5 | Yes | True | Yes | (0.152, 0.980) |

**Findings:**

* The training sample size played a crucial role in determining the model's learning characteristics and its performance on unseen data.
* Although RMSprop is one of the best optimizer functions for building convolutional neural networks, Adam has the upper edge due to its unique combination of using Momentum and RMSprop together to optimize the neural network effectively.
* Since pre-trained networks are trained on a broad range of image categories, freezing the initial layers of the model retains the general categorization capabilities. This approach forces the model to focus specifically on the image recognition task at hand. This strategy proved effective, as seen in the fine-tuned models Model 5, which achieved the highest accuracy within their respective sample sizes.

# Final Comments:

Even with pre-trained networks, there is a significant relationship between sample size and model performance. When the sample size increased from 1000 to 3000, there was a noticeable improvement in the model's accuracy and a reduction in loss. Providing more data, along with augmented images, proved to be an efficient way to boost the model's performance.

Freezing the initial layers of the pre-trained network and preventing it from updating its weights during training effectively controlled overfitting and helped the model generalize better on unseen data.

