CNN Experimentation on Mini-imagenet Dataset

Divya Jyoti

22 June 2025

1. Introduction

This report documents the process and results of experimenting with various **Convolutional Neural Network (CNN)** architectures on a custom image classification dataset. The primary objective is to **evaluate and compare the performance of different model configurations** under controlled settings. The experiments are implemented using **PyTorch,** and the notebook includes detailed code, training procedures, and evaluation metrics.

2. Data Preparation

The dataset was unzipped and loaded using torchvision.datasets.ImageFolder. **Data augmentation** techniques such **as resizing, random cropping, and horizontal flipping** were applied to the training images. Both training and test sets were normalized using ImageNet mean and standard deviation values.

3. Model Architecture

A flexible CNN model class `CustomCNN` was defined, allowing customization of **convolution layers, fully connected layers, stride, pooling, and filter** sizes. The architecture adapts its fully connected input size dynamically.

4. Training and Evaluation

The model was trained using Adam optimizer and CrossEntropy loss. Accuracy was computed using sklearn.metrics. Misclassified images were collected for visual inspection.

5. Experiments

Several experiments were run with different combinations of convolution layers, fully connected layers, filter sizes, and epochs. The goal was to compare the impact of depth and architectural complexity on model performance.

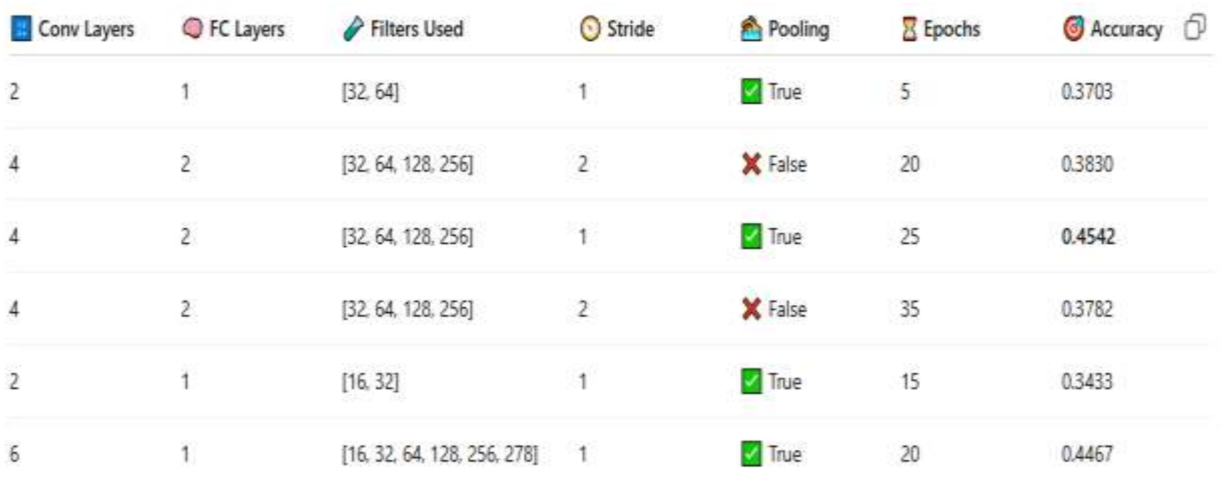
6. Results

The experiment results were summarized as follows:

We conducted several experiments to evaluate how different CNN architectures affect classification accuracy. The table below summarizes each configuration and its corresponding test accuracy.

Results Table:

"Model Accuracy without Data Augmentation and Occlusion-Based Training"



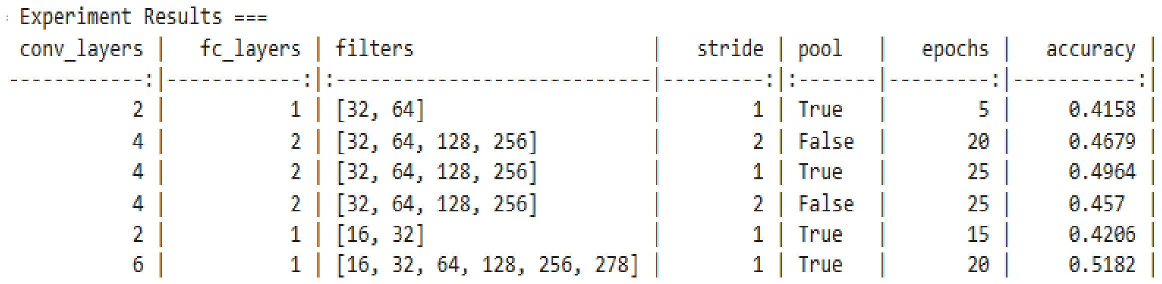
🟦 Conv Layers	🔴 FC Layers	🖊 Filters Used	⏱ Stride	🏊 Pooling	⏳ Epochs	🎯 Accuracy
2	1	[32, 64]	1	✅ True	5	0.3703
4	2	[32, 64, 128, 256]	2	❌ False	20	0.3830
4	2	[32, 64, 128, 256]	1	✅ True	25	**0.4542**
4	2	[32, 64, 128, 256]	2	❌ False	35	0.3782
2	1	[16, 32]	1	✅ True	15	0.3433
6	1	[16, 32, 64, 128, 256, 278]	1	✅ True	20	0.4467

Observations

- The **best accuracy (0.4542)** was achieved with:
  - 4 convolutional layers
  - 2 fully connected layers
  - Moderate stride of 1
  - Max pooling enabled
  - 25 training epochs
- Very deep architectures (like 6 conv layers) performed well but slightly below the best.
- Small networks (e.g., 2 conv layers) underperform due to limited feature extraction capacity.
- **Pooling improves generalization** when used with stride 1. Stride 2 without pooling generally underperforms.

Training with **data augmentation techniques** such as random crop-resize, horizontal flip, and occlusion simulation led to **improved model performance**, as demonstrated in the results shown in the table below.

"Model Accuracy with Data Augmentation and Occlusion-Based Training"



```
 Experiment Results ===
 conv_layers |   fc_layers | filters                          |   stride | pool    |   epochs |   accuracy |
------------:|------------:|:---------------------------------|---------:|:--------|---------:|-----------:|
           2 |           1 | [32, 64]                         |        1 | True    |        5 |     0.4158 |
           4 |           2 | [32, 64, 128, 256]               |        2 | False   |       20 |     0.4679 |
           4 |           2 | [32, 64, 128, 256]               |        1 | True    |       25 |     0.4964 |
           4 |           2 | [32, 64, 128, 256]               |        2 | False   |       25 |     0.457  |
           2 |           1 | [16, 32]                         |        1 | True    |       15 |     0.4206 |
           6 |           1 | [16, 32, 64, 128, 256, 278]      |        1 | True    |       20 |     0.5182 |
```

The table summarizes the classification performance across different CNN configurations trained using **data augmentation and occlusion simulation**. Key architectural parameters such as number of convolutional and fully connected layers, filter depth, pooling, stride, and training epochs were varied to analyze their impact.

Key Observations:

- **Deeper networks yield better accuracy**:

  The best result (accuracy = **0.5182**) was achieved using a **6-layer convolutional network** with progressively increasing filters [16, 32, 64, 128, 256, 278] and a single fully connected layer.

  This indicates that deeper architectures **capture hierarchical visual features more effectively**, especially with augmentation.

- **Pooling improves generalization**:

  Models with **pooling enabled (**pool=True**)** consistently performed better than their non-pooled counterparts.

  E.g., Row 3 vs. Row 4: Both have same filters and epochs, but **pooled version (Row 3)** achieved **0.4964**, while **non-pooled (Row 4)** only got **0.457**.

- **Impact of filter complexity**:

  Shallow models with fewer filters (e.g., [16, 32] in Row 5) underperformed (**0.4206**) compared to deeper filter stacks like [32, 64, 128, 256] or beyond.

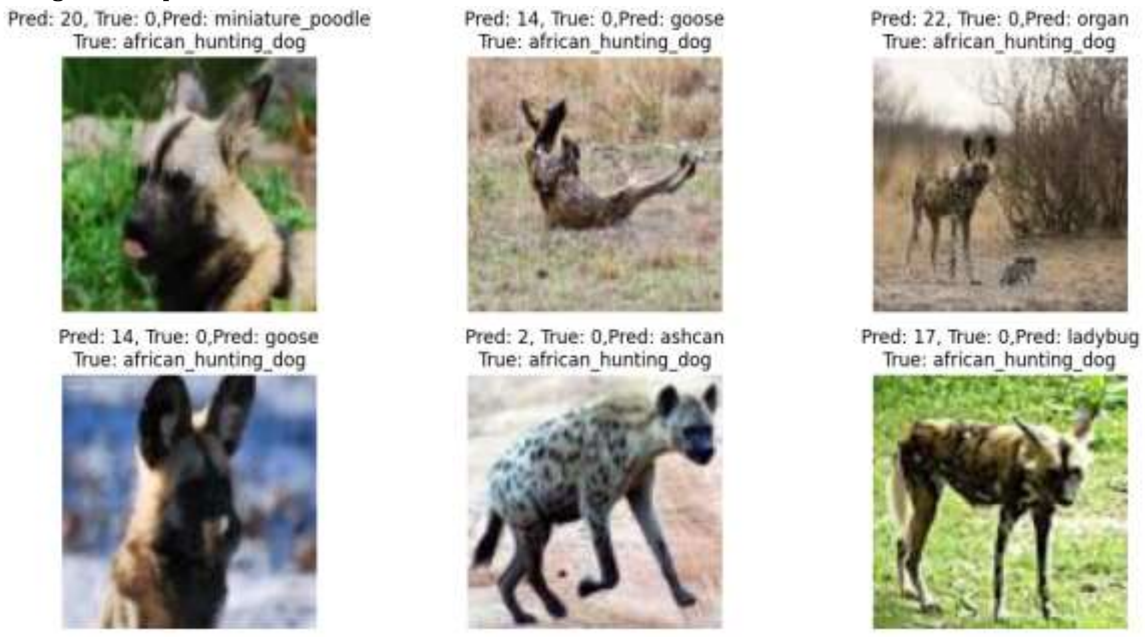
  Larger and deeper filter stacks appear to **extract richer feature representations**.

- **Epoch count and accuracy**:

  More training epochs generally led to better accuracy up to a point (e.g., 25 epochs in Row 3 with **0.4964**).

7. Misclassified Samples

The model's predictions were visually inspected by plotting a sample of misclassified images with predicted and true labels.


Pred: 20, True: 0,Pred: miniature_poodle
True: african_hunting_dog
Pred: 14, True: 0,Pred: goose
True: african_hunting_dog
Pred: 22, True: 0,Pred: organ
True: african_hunting_dog
Pred: 14, True: 0,Pred: goose
True: african_hunting_dog
Pred: 2, True: 0,Pred: ashcan
True: african_hunting_dog
Pred: 17, True: 0,Pred: ladybug
True: african_hunting_dog

8. Occlusion Analysis

We performed occlusion sensitivity analysis on a sample image to better understand the model's decision-making process.

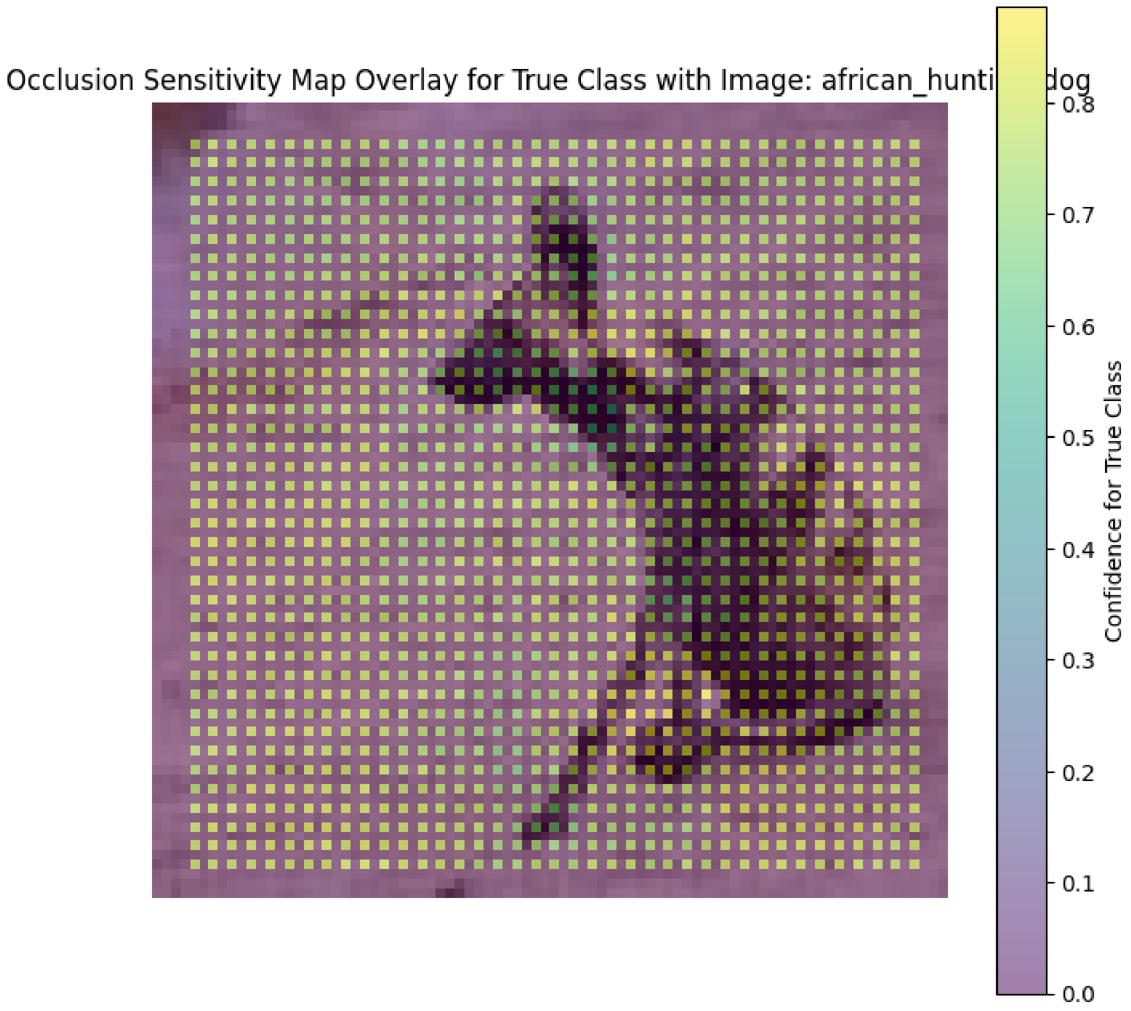
- The **occlusion sensitivity map** visually highlights the regions of the image that are most influential in the model's correct classification.

- **Darker regions** (in the 'viridis' colormap) indicate areas where occluding a patch leads to a significant **drop in confidence** for the true class. These regions likely contain **critical features** that the model relies on.

- In contrast, **lighter areas** reflect regions where occlusion has **minimal impact** on confidence, suggesting they are **less important** for the model's prediction.

- By overlaying the sensitivity map on the original image, we can correlate important areas with actual visual features (such as object parts), offering insights into the **visual cues** the model uses to make its decisions.

Occlusion Sensitivity Map for a Correctly Classified Sample

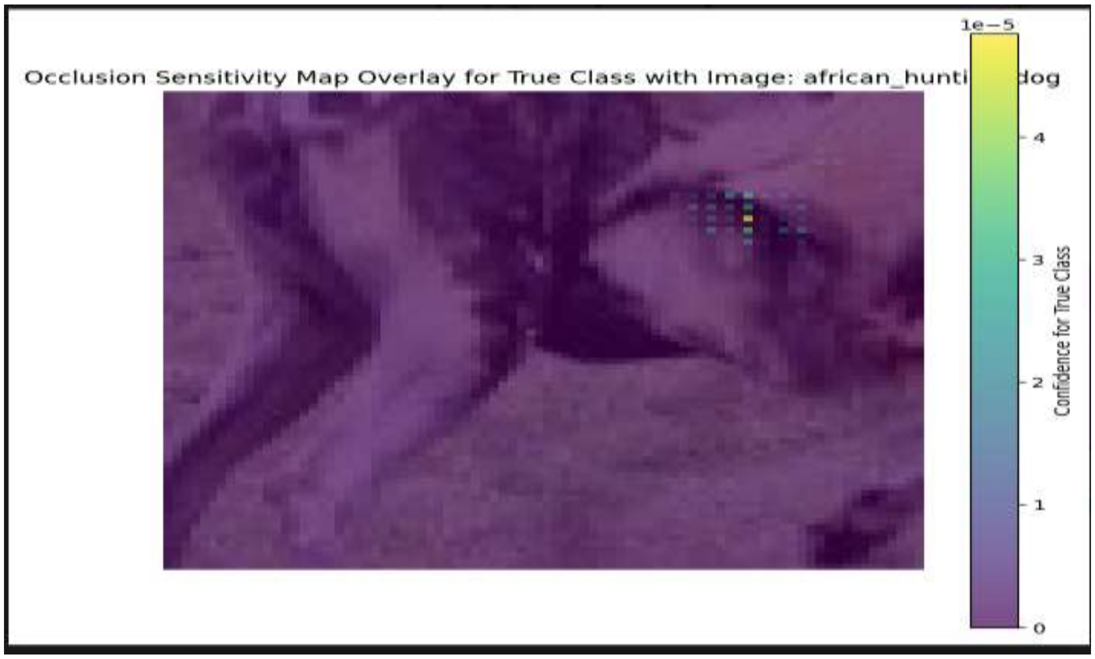


Above occlusion sensitivity map corresponds to a correctly classified image labeled as **"african_hunting_dog"**, with the model achieving a **confidence score of approximately 0.8** for the true class.

- The map reveals that the **head, torso, and leg regions** are critical to the model's decision — occluding these parts leads to a noticeable drop in confidence (shown in darker shades).
- Conversely, **occluding the background** or peripheral areas has minimal impact, indicating that the model **focuses on semantically meaningful features**.
- This suggests that the model has learned to **reliably attend to object-relevant regions**, which contributes to the correct and confident prediction.

Occlusion Sensitivity Map of a Misclassified Sample:

This image represents an occlusion sensitivity map for a **misclassified example** of the `african_hunting_dog` class.



Key Observations

- The model demonstrates **very low confidence** in the true class across the entire image, as reflected by near-zero values in the occlusion sensitivity map.
- **Occluding different regions** has **minimal impact** on the model's prediction, indicating that it is **not leveraging meaningful features** relevant to the target class.
- A **small area near the head or face** causes a slight confidence drop when occluded, but not enough to correct the prediction.

Interpretation

The misclassification may be attributed to:

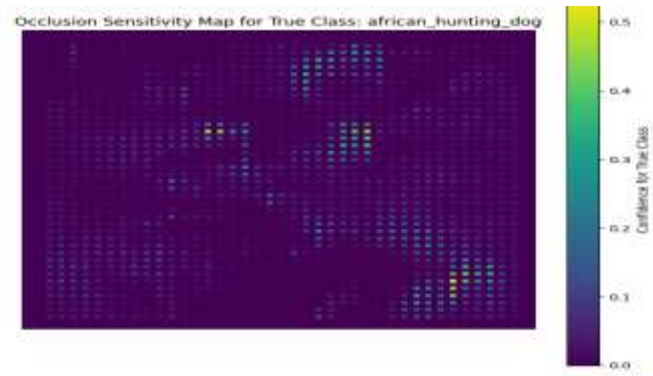
- **Ambiguous or weak visual features** in the image, or overall poor image quality.
- **Underfitting** or lack of sufficient exposure to similar examples during training.

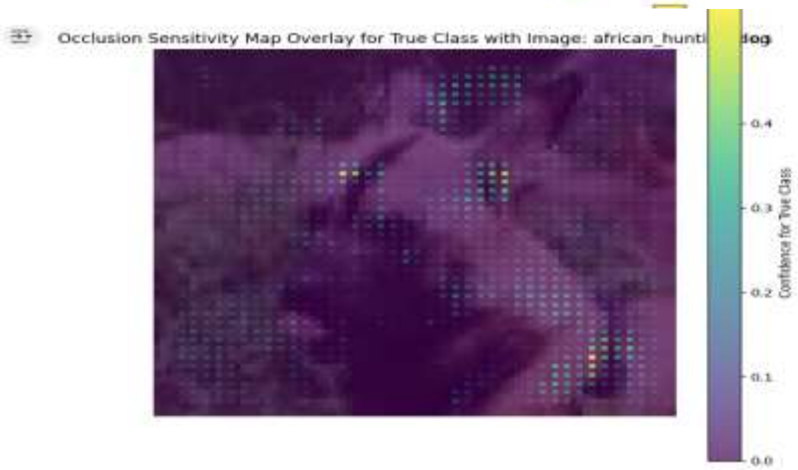
- **Confusion with visually similar classes** that share overlapping features.

These findings suggest a need for:

- **Greater diversity in training data**,
- Possibly **deeper or more expressive model architectures** to capture finer-grained features.

Occlusion Sensitivity Map of a random data from test set





Observation Based on the Above Image

- In this case, the **darker regions** primarily reflect the model's **already low confidence** in the true class, rather than a significant drop caused by occlusion.
- Only a **few scattered patches** display green or yellow, indicating **minimal sensitivity** to occlusion. This suggests that the model's prediction is **generally uncertain**.
- The lack of concentrated high-sensitivity regions implies that the model is **not strongly relying on any specific part of the image**, which is characteristic of either a **misclassification** or a **low-confidence correct prediction**.

9. Conclusion

This project explored the design, training, and evaluation of Convolutional Neural Network (CNN) models for image classification using a custom dataset. Various experiments were conducted to assess model performance under different configurations, including:

- **Baseline models vs. augmented training**
- Use of **data augmentation techniques** such as random cropping, flipping, and occlusion simulation
- Application of **occlusion sensitivity analysis** to interpret model behavior
- **Varying architectural depth**, number of filters, and training epochs

Key insights and findings:

- **Data augmentation significantly improved accuracy and generalization**, especially when occlusion simulation was included during training.
- **Experimentation with deeper architectures** and increased epochs yielded **higher accuracy**, particularly when combined with appropriate **filter sizes and pooling strategies**.
- Occlusion sensitivity maps provided valuable interpretability, revealing how confidently and accurately the model relies on specific image regions for its predictions.
- Correctly classified samples with high confidence showed **focused attention** on meaningful object parts, while misclassified or low-confidence predictions exhibited **diffused or irrelevant attention**, highlighting areas for model improvement.
- **Misclassification analysis** revealed that certain classes were **more frequently confused**, likely due to **strong visual similarity**, emphasizing the importance of class-level feature separation.
- Comparative evaluation across models helped identify architectural and training choices that enhanced both **robustness** and **interpretability**.

Overall, this study demonstrates that comparing CNN models with varying architectural parameters, augmenting training data, increasing model depth and capacity, and performing interpretability and error analysis are all essential steps toward building reliable, accurate, and explainable deep learning models for image classification.