Building a Convolutional Neural Network for Image Classification

Student ID: 23038567

GitHub Link: https://github.com/Vaishali8567/Machine-Learning---CNN-Tutorial

Dataset Link: https://data.caltech.edu/records/mzrjq-6wc02

I. Introduction

The aim of this tutorial is to provide a comprehensive guide on machine-learning technique: Convolutional Neural Networks (CNN).

This tutorial caters to those looking to gain understanding on CNNs practically, focusing on dataset preparation, model training, and evaluation. By the end of this tutorial, readers will:

- Learn how CNNs work and why they excel on tasks involving images.
- Discover the significance of appropriate dataset preparation, which includes data augmentation and preprocessing.
- Gain insights on how use metrics like accuracy and confusion matrices to assess a CNN model's performance.

II. Convolutional Neural Networks (CNNs)

Overview

- Convolutional Neural Networks (CNNs) are a specialized type of neural network designed primarily for processing structured data like images.
- CNNs have completely transformed the field of computer vision by enabling significant improvements in image recognition, object detection, and image segmentation.
- They eliminate the need for intensive human feature engineering by automatically extracting hierarchical features from raw input data.

Key Components

i. *Convolutional Layers:* The foundation of CNNs is the convolutional layer. In order to extract spatial features like edges, textures, or patterns, it applies a collection of learnable filters, also known as kernels, to the input image. These filters are dragged over the input, and dot products between the filter values and the areas of the image that overlap are calculated.
   - Key Attributes*:* Filter size, stride (step size for sliding), and padding (handling edge regions).
   - Output: Produces feature maps that show the input image's identified features.

ii. *Pooling Layers:* By reducing the spatial dimensions of feature maps, pooling layers guarantee a decrease in overfitting and computational expense while preserving the most relevant data.
   - Types: Max Pooling, which keeps the maximum value from each region and Average Pooling, which calculates the average of the values in each region.
   - Output: Smaller feature maps with retained key features.

iii. *Fully Connected Layers (Dense Layers):* Fully connected layers connect each neuron in one layer to every other layer's neuron. By merging the features retrieved by the convolutional and pooling layers and projecting them to the output classes, they function as classifiers.
   - Key Role: Converts the spatially localized features into classification class scores.

III. Dataset

For this tutorial, a custom CNN architecture was developed to classify images from the **Caltech-101**.

- Caltech-101 dataset is one of the popular benchmark datasets in computer vision.
- Images are divided into 101 item classes, including guitars, elephants, and airplanes, in addition to one "background" category. There are between 40 to 800 photos in each category, varying in size and resolution.

IV. Why use CNN for this Dataset?

CNNs are particularly well-suited for this dataset for the following reasons

- **Feature Extraction Capability:** CNNs are made to effectively and automatically extract spatial hierarchies of features, including patterns, textures, and edges, from images. This is advantageous for the Caltech-101 dataset since objects in each category have unique visual patterns that CNNs can learn to identify.
- **Translation Invariance:** The set contains objects in a variety of scales, orientations, and locations. CNNs may focus on pertinent characteristics regardless of where they are in the image since they use pooling layers to achieve translation invariance.
- **Scalability:** CNNs offer a scalable method for multi-class classification jobs because of the comparatively large number of categories (102, including the background).
- **Generalization:** CNNs are a good option for addressing this since they can manage and generalize the visual diversity of Caltech better thanks to approaches like data augmentation and dropout.

V. Preprocessing Dataset

Preprocessing is a critical step to ensure that the dataset is in the correct format and scale for the CNN. The following steps were implemented:

- **Dataset Split:** In deep learning models like CNNs, dataset is divided into subsets to ensure that the model is learning generalizable patterns. It is typically split into:
  - Training Set: This subset enables the model to use backpropagation to optimize weights and discover patterns in the data.
  - Validation Set: This subset is used to assist hyperparameter adjustment and avoid overfitting by assessing the model's performance during training.
  - The dataset was divided into two parts: 20% for validation and 80% for training.
  - This ratio strikes a balance between the necessity for reliable validation and enough training data.
- **Images Resizing**
  - All TensorFlow preprocessing tools were used to compress all pictures to a standard 150x150 pixel size.
  - This lowers the computing requirements and guarantees consistency in input dimensions.
- **Normalization**
  - Pixel values were normalized to the range [0, 1] by dividing the pixel intensity values by 255. By scaling input characteristics, this guarantees faster convergence and helps in training process.

- **Data Augmentation**
  - To improve generalization and prevent overfitting, data augmentation techniques like random flips, rotations, zooming and brightness adjustments were performed using Keras's *ImageDataGenerator*.
- **Shuffling and Batching**
  - To maximize compute and memory utilization, training and validation sets were handled in batches of size 32.

VI. Build CNN Model

'Sequential' class is a simple way to build a neural network layer-by-layer in linear stack. The output of one layer immediately feeds into the next, and each layer is added in turn. This model includes:

- **Convolutional Layers**
  - To identify patterns ranging from edges to complex shapes, three layers with 32, 64, and 128 filters each are used.
  - Max-pooling lowers spatial dimensions to preserve important characteristics while avoiding overfitting, while ReLU activation adds non-linearity.
- **Flattening**
  - Convolutional layers' multidimensional outputs are transformed into a 1D vector so that fully connected layers can use them.
- **Fully Connected Layers**
  - A dense layer of 128 neurons uses ReLU activation in combination with features for complicated interactions.
  - A dropout layer (50%) to reduce overfitting.
  - The final dense layer (102 neurons) with *Softmax* activation generates class probabilities.
- **Compilation**
  - For effective training, the model makes use of categorical cross-entropy loss and the Adam optimizer.

VII. Train the Model

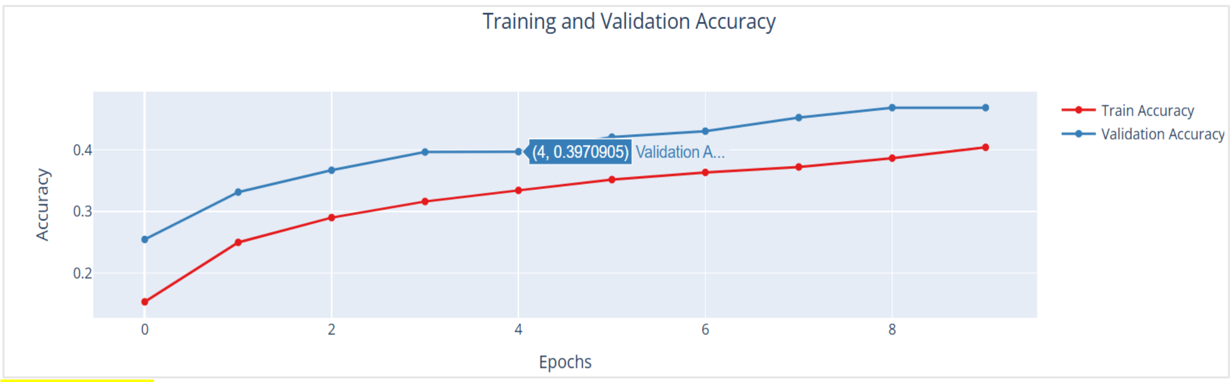
To reduce the error in its predictions, the model modifies its internal parameters (filters and weights) throughout training. Depending on the network's depth, the model should be able to automatically identify patterns and features in the data, such as edges, textures. In this tutorial:

- The CNN is trained on the data using the model.fit() function.
- This model is trained for maximum of 10 epochs. However, training may end sooner if the criteria of the callbacks are satisfied.

VIII. Training and Validation Performance

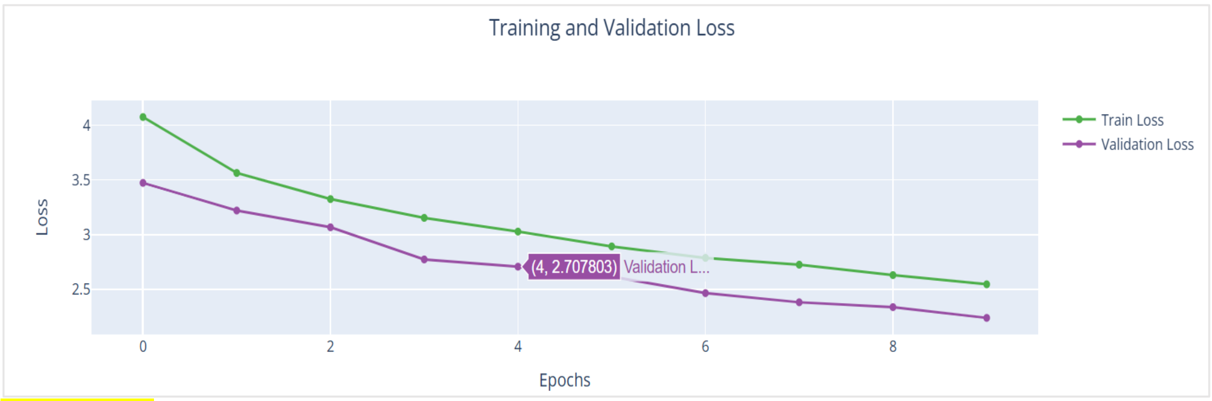
"Accuracy" and "Loss" were the two main metrics used to evaluate the model's performance throughout training and validation.

- **Accuracy**:
  - The proportion of correctly categorized images relative to the total number of images is known as Accuracy.
  - The model's ability to generalize to new data was revealed via validation accuracy.



Observation: A plateau was approached by the training and validation accuracy curves, indicating that the model had reached its maximum potential.

- **Loss:**
  - The discrepancy between the actual labels and the anticipated outputs is represented by Loss.
  - While an increase signalled possible overfitting, a drop in validation loss indicated better model performance.



Observation: A slight degree of overfitting was suggested by the difference between training and validation loss, which might be reduced by using strategies like dropout.

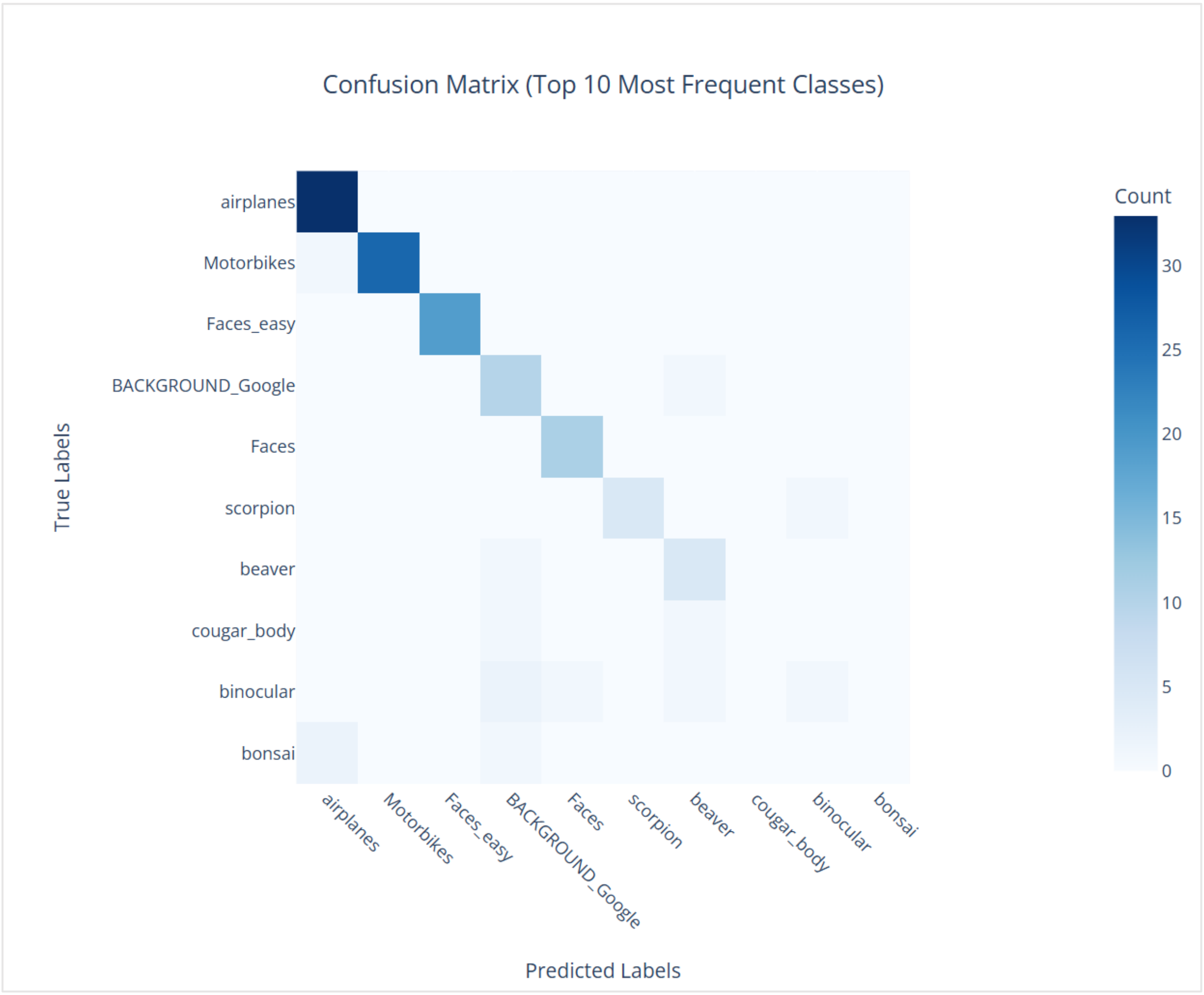
IX. Model Evaluation

Following CNN training, the model's capacity for generalization was assessed using the unseen test set.

- **Test Accuracy**
  - The ratio of correctly identified images to the total number of images in the test set.
  - The model successfully trained to categorize the various classes in the Caltech-101 dataset, as indicated by a test accuracy.
- **Classification Report**
  - In order to better understand how well the model performed on unbalanced classes, additional metrics were calculated:
    - Precision: The percentage of positive samples that were correctly recognized.
    - Recall: The model's capacity to include all pertinent positive samples.
    - F1 Score: The harmonic means of recall and precision, which balances the two measures.

Confusion Matrix

- A simplified confusion matrix was created for the top ten most common classes in the dataset due to the vast number of classes.
- This method concentrated on the most significant predictions while making the visualization easier to understand.
- Analysis of the confusion matrix revealed which classes have a high rate of misclassification.
- Check to see if the errors were concentrated in visually comparable groups.



Observations:

- Relatively strong performance is displayed by classes such as "airplanes," "motorbikes," indicating that the model has mastered the ability to recognize distinctive characteristics for these groups.
- When non-zero values are present in off-diagonal elements, misclassifications are seen. For example, some "Faces_easy" samples might have been mistakenly identified as "Faces," suggesting confusion brought on by visual similarities.
- Inter-class misunderstanding is seen in classes like "Faces_easy," "Faces," and "Background_Google." This suggests that the model may have trouble differentiating between these categories due to visual similarity or overlapping features.

X. Advantages of CNN

- Automated Feature Learning: CNNs do not require human feature engineering because they automatically extract pertinent features from data.
- Robust Performance: They perform well on a variety of image tasks because they can adapt to changes in noise, occlusions, and lighting.
- End-to-End Training: CNNs enable the optimization of feature extraction and classification in a single, integrated process.
- Adaptability: Training time and resources can be minimized by fine-tuning pre-trained models, such as VGG and ResNet, for particular workloads.
- Broad Applicability: CNNs are employed for computer vision tasks such as object identification and segmentation in addition to picture classification.

XI. Limitations of CNN

- Large Data Requirements: CNNs can perform poorly on small datasets and require enormous volumes of labeled data.
- Computational Intensity: CNN training uses a lot of resources and calls for powerful GPUs or TPUs.
- Overfitting: On limited datasets, complex models may overfit, necessitating regularization strategies like dropout.
- Interpretability: CNNs are challenging to understand because their decision-making process can often become opaque.
- Sensitivity to Adversarial Attacks: CNNs are susceptible to making inaccurate predictions when input pictures undergo subtle, undetectable alterations.

XII. Conclusion

Using the **Caltech-101 dataset**, this tutorial illustrates how Convolutional Neural Networks (CNNs) can be used for picture classification. The methodical methodology offers insights into CNN design and performance visualization while outlining the crucial phases, from dataset preparation to model training and evaluation.

In addition to teaching the basic procedures, this lays the groundwork for future research.

Key Findings:

- Model Performance: On the validation set, the CNN's accuracy was moderate. The model did well on classes with distinguishing characteristics but it had trouble with fine-grained categorization among visually similar classes, according to the confusion matrix.
- Challenges: Inter-class misunderstanding and misclassifications demonstrated the model's shortcomings in feature extraction for complicated or ambiguous categories.
- Improvements: Methods like hyperparameter tuning, pre-trained models (like transfer learning), can improve performance even more and deal with issues that arise.

XIII. References

1. Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville, MIT Press.
2. Very Deep Convolutional Networks for Large-Scale Image Recognition, arXiv:1409.1556, Simonyan, K., & Zisserman, A.
3. Blog: "A Comprehensive Guide to CNN Architectures" - Towards Data Science (Medium).
4. TensorFlow Documentation: https://www.tensorflow.org/
5. Keras Documentation: https://keras.io/