Pokemon Image Classifier using CNN

Diego Perdomo Salcedo

Abstract –

The purpose of this report is to analyze a model developed to classify images of Pokémon into their respective classes. This document discusses the structure and approach taken in developing the model, highlighting its architecture, data preprocessing, model performance, and areas for future improvement. Through training, the model achieved an average test accuracy of 60%, showcasing its potential as a tool for automatic Pokémon classification.

1. INTRODUCTION

In today's world, image classification has applications far beyond entertainment, helping industries tackle complex identification tasks across various domains. Pokémon classification presents a unique challenge due to the sheer diversity in Pokémon appearances, and a successful model here can demonstrate the potential for applying this technology in real-world scenarios.

Automating the classification of Pokémon images offers a fun, yet powerful, framework that could be repurposed in fields such as environmental science, where similar classification models could be used to identify and categorize animal species or plants based on visual input alone. Such a model could support wildlife research by identifying endangered species, monitoring population dynamics, and even assisting in conservation efforts through efficient and accurate classification.

The approach taken in this project involves building a convolutional neural network (CNN) to classify 150 different Pokémon classes based on their unique visual characteristics. The model aims to achieve high accuracy by learning the distinguishing patterns between classes. This model could serve as a foundational structure for broader applications. Whether it's classifying animals, plants, or other image-heavy data in research and beyond. This project underscores how image classification can transition from popular culture to practical applications, potentially impacting fields as diverse as ecology, zoology, and agriculture.

2. DATA CLEANING AND PROCESSING

The Pokémon dataset comprises images categorized by class (different Pokémon). Each image was resized to a standard dimension of 128x128 pixels for consistency and normalized to scale pixel values between 0 and 1, which facilitates smoother training of the CNN model. Data augmentation techniques, such as rotations and flips, were applied to diversify the training set, effectively expanding the dataset and making the model more robust.

Since the dataset originally contained an imbalance in the number of images per class, we applied stratified sampling to ensure equal representation. This balanced approach mitigates bias toward more frequent classes.

3. DATASET SPLIT

To achieve effective model evaluation, the dataset was divided into training (70%), validation (15%), and test (15%) sets. The training set enables the model to learn the features of each Pokémon class, the validation set assists in model tuning, and the test set evaluates its generalization ability. Stratified sampling ensured an even distribution across these sets.

4. MODEL ARCHITECTURE

The convolutional neural network (CNN) architecture was chosen for this Pokémon classification task due to its effectiveness in handling image data.

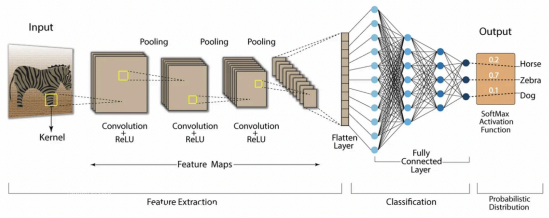


Figure 1: CNN

4a.-Framework

This CNN model is constructed using the **TensorFlow** and **Keras** frameworks, which provide powerful tools for defining, training, and optimizing deep learning models. Keras, a high-level API within TensorFlow, simplifies the process of building and managing layers, allowing focus on architectural design rather than on low-level computations.

TensorFlow's Keras API allows for efficient model training by managing backpropagation, gradient computation, and optimizations. By using this framework, complex computations and model configurations are streamlined, making it easier to implement and evaluate the model.

4b.-CNN Model Components

Input Layer: The input layer is defined to accept 128x128 RGB images (3 color channels). This layer acts as the model's entry point, feeding image data into the network for feature extraction.

Convolutional Layers: Convolutional layers are responsible for identifying visual patterns in the images.

- In each convolutional layer, a set of filters (or kernels) is applied to the input to create feature maps. For instance, the first convolutional block has 32 filters, followed by 64 and 128 in subsequent blocks.
- As the network goes deeper, more filters are added, capturing increasingly complex features. Early layers detect edges and textures, while later layers capture higher-level features such as shapes and patterns.

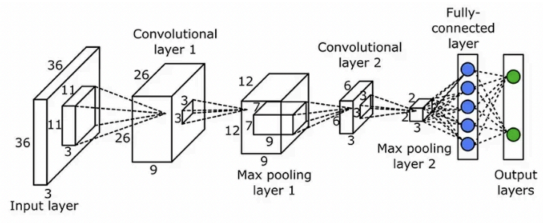


Figure 2: Convolutional Layer

Activation Layer (RELU): Each convolutional layer in the CNN model is followed by a ReLU (Rectified Linear Unit) activation function, an essential component that introduces non-linearity to the network, enabling it to learn complex patterns**.**

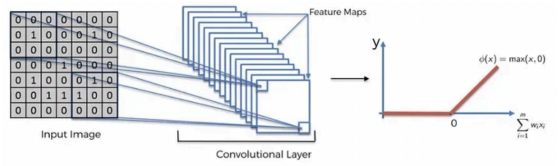


Figure 3: Activation Layer

Batch Normalization: Batch normalization normalizes the output of a layer, stabilizing and accelerating the training process by reducing internal covariate shift.

- This layer adjusts and scales activations, allowing the model to converge faster and helping to reduce overfitting by providing regularization.

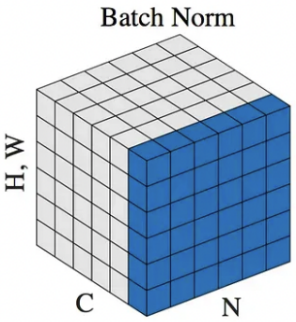


Figure 4: Batch Normalization

Max-Pooling Layers: Max-pooling reduces the spatial dimensions (height and width) of the feature maps while retaining the most salient information.

- By using a 2x2 filter, each max-pooling layer downsamples the feature maps, which reduces computational complexity and helps prevent overfitting by generalizing features.

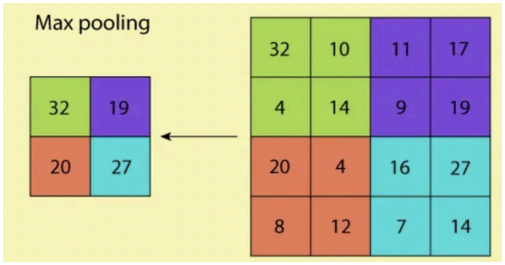


Figure 5: Max Pooling

Dropout Layers: Dropout layers randomly deactivate a fraction of neurons during training, making the model less sensitive to specific neurons and helping prevent overfitting.

- In the first convolutional blocks, a 25% dropout is used, while in the fully connected layer, it is increased to 50%. This helps the network generalize better to new data.

Fully Connected Layer: After the convolutional and pooling layers, the feature maps are flattened and passed through a dense (fully connected) layer with 512 neurons.

- This layer combines the features learned by the convolutional layers, enabling the model to learn complex associations before passing the data to the output layer.

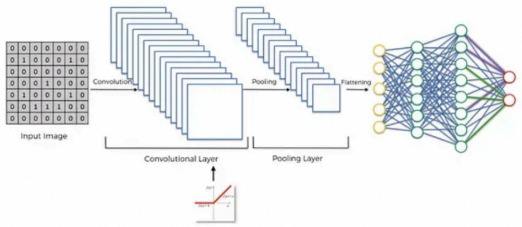


Figure 6: Dense Layer

Output Layer: The final layer is a softmax layer, which outputs a probability distribution across the possible Pokémon classes.

- Softmax ensures that the model's predictions are interpretable as probabilities, allowing for classification of the input image into one of the Pokémon classes.

This architecture is well-suited for image classification, as the hierarchical layers capture both low- and high-level features, enabling the model to distinguish subtle differences between classes.

5. MODEL TRAINING AND EVALUATION

The model was trained using a categorical cross-entropy loss function, optimized with the Adam optimizer, and monitored for accuracy. Training was conducted over 10 epochs, with early stopping based on validation loss to prevent overfitting.

During training, both training and validation accuracy were monitored. The validation set provided a gauge of the model's generalization to unseen data, while a learning rate scheduler helped fine-tune the convergence rate.

The final results ended with a training accuracy of 99%, validation accuracy of 61%, and a test accuracy of 60%. The final test set accuracy reflects the model's capability to generalize across Pokémon types it has not seen before.

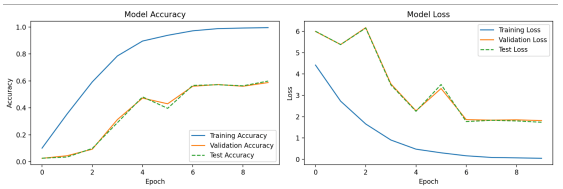


Figure 7: Accuracy and cost over epoch

The very high training accuracy indicates that the model fits the training data extremely well. However, this could be a sign of overfitting if the model has learned the training data too specifically, capturing even minor variations or noise in the data. In other words, the model exhibits low bias on the training set since it can approximate the training data with high accuracy.

The significant drop in accuracy on the validation (61%) and test sets (60%) reveals high variance. This variance suggests that the model struggles to generalize to unseen data, which is characteristic of an overfit model. The gap between the training and

validation/test accuracies implies that the model has not learned generalizable patterns but has instead memorized details specific to the training set, increasing the bias on the validation and test sets.

6. FUTURE IMPROVEMENTS

The model's low bias on the training data (high training accuracy) combined with high variance (large accuracy gap between training and validation/test) indicates that the model is indeed overfitting.

6a.-REGULARIZATION

- L2 Regularization: Adding L2 regularization could help reduce the complexity of the model by penalizing large weights, encouraging the model to learn more generalized features.
- Dropout: Increasing dropout rates can help the model focus less on specific neurons, forcing it to learn more generalized features across the dataset. However, there's a balance, as too much dropout can lead to underfitting.

6b.-MODEL COMPLEXITY

To improve classification accuracy further, particularly for challenging classes, we could increase model complexity by incorporating additional convolutional layers or experimenting with larger networks like EfficientNet.

6c.-CROSS-VALIDATION

Using cross-validation techniques, such as k-fold cross-validation, can give a more robust evaluation of model performance and potentially reduce overfitting by training on multiple subsets of the data.

6d.-DATA AUGMENTATION

Generating new variations of images by applying transformations (like rotation, zoom, brightness adjustments, and cropping) can help the model learn more robust, generalized features.

7. IMPROVED MODEL

After analyzing the performance issues in the initial model, several strategic improvements were implemented to enhance the model's performance and address the overfitting problem. The improved model incorporates architectural modifications and training optimizations designed to achieve better generalization while maintaining strong learning capabilities.

7a. ARCHITECTURE MODIFICATIONS

The architecture of the improved model maintains the core CNN structure while introducing several key enhancements:

Input Processing: A significant addition was the implementation of input normalization through a rescaling layer directly in the model architecture. This ensures all images are consistently processed during both training and inference phases, promoting more stable learning patterns.

Convolutional Layer Enhancement: The convolutional layers were modified to include 'same' padding, preserving spatial dimensions throughout the network. This modification helps maintain important spatial information that might otherwise be lost at image borders, particularly crucial for detecting Pokemon features that may extend to the edges of images.

Progressive Dropout Implementation: Instead of using a fixed dropout rate of 25% across all convolutional layers, the improved model implements a progressive dropout strategy:

- Initial convolutional blocks: 10% dropout
- Final convolutional block: 20% dropout
- Dense layer: 30% dropout

This graduated approach helps maintain critical information in the early layers while preventing overfitting in deeper layers where feature representations become more specialized.

Dense Layer Optimization: The fully connected layer was optimized by reducing its size from 512 to 256 neurons. This reduction in parameters helps combat overfitting while maintaining sufficient capacity for learning complex Pokemon features.

7b. TRAINING ENHANCMENTEST

The training process was refined with several important modifications:

Learning Rate Management: The improved model implements a more controlled learning rate strategy, starting with a standard initial rate of 0.001. This provides a better balance between learning speed and stability during the training process.

Data Augmentation Integration: Real-time data augmentation was introduced during the training phase, including:

- Horizontal flipping of images
- Random rotations up to 10 degrees
- Random zoom adjustments up to 10%

These augmentations effectively increase the diversity of the training data, helping the model learn more robust and generalizable features.

Training Duration and Monitoring: The training process was extended from 10 to 20 maximum epochs, allowing more time for proper convergence. However, this was balanced with more stringent early stopping criteria:

- Reduced patience from 10 to 5 epochs
- Implementation of a minimum improvement threshold (delta) of 0.001
- More aggressive monitoring of validation loss to prevent overfitting

7c. ARCHITECTURE COMPARISON

The architectural changes can be observed in the layer specifications:

Original Model:

- Basic convolutional layers without padding
- Fixed 25% dropout rate across all layers
- Larger dense layer with 512 neurons
- Basic training process without data augmentation

Improved Model:

- Padded convolutional layers for spatial preservation
- Progressive dropout strategy (10% -> 20% -> 30%)
- Optimized dense layer with 256 neurons
- Integrated data augmentation pipeline

These modifications were done to address the overfitting issues observed in the original model while maintaining the network's capacity to learn meaningful features from the Pokemon dataset.

8. IMPROVED MODEL RESULTS

The implementation of the architectural and training modifications detailed in the previous section led to significant improvements in both model performance and stability. A detailed analysis of the results reveals substantial progress in addressing the overfitting issues while maintaining strong learning capabilities.

8a. TRAINING DYNAMICS ANALYSIS

The improved model exhibited markedly different training characteristics compared to the original implementation. While the original model showed rapid convergence to high training accuracy (98% by epoch 8), the improved model demonstrated a more controlled learning progression, achieving 90% training accuracy by epoch 18. This more gradual learning curve suggests better regularization and more robust feature learning.

The validation accuracy also showed notable improvements. The original model's validation accuracy fluctuated between 39% and 58%, indicating unstable learning. In contrast, the improved model maintained a steadier progression, reaching 68% validation accuracy. This more stable behavior suggests the model was learning generalizable features rather than memorizing training data.

8b. PERFORMANCE COMPARISON

The final metrics demonstrate significant improvements across key indicators:

Original Model:

- Training Accuracy: 98%

- Validation Accuracy: 58%
- Test Accuracy: 60%
- Training-Validation Gap: 40%

Improved Model:

- Training Accuracy: 90%
- Validation Accuracy: 68%
- Test Accuracy: 66%
- Training-Validation Gap: 22%

The reduction in the training-validation accuracy gap from 40% to 22% indicates much better generalization capabilities. The test accuracy improvement from 60% to 66% represents a significant 10% relative increase in performance on unseen data.

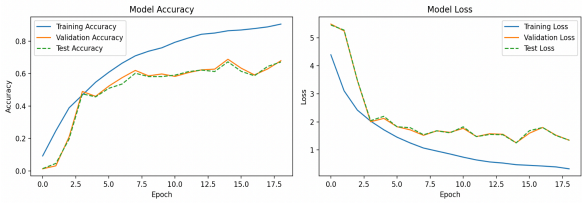
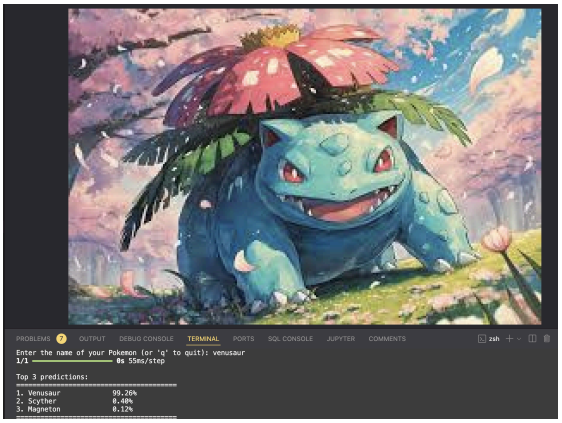


Figure 8: Improved model's Accuracy and cost over epoch

The model is also able to classify more complex images compared to the previous model.



9. CONCLUSIONS

The Pokemon classification project demonstrates significant advancement in automatic image classification, evolving from an initial model with notable overfitting issues to an improved implementation with better generalization capabilities. Through systematic improvements in architecture and training methodology, the model progressed from 60% to 66% test accuracy, while significantly reducing the training-validation accuracy gap from 40% to 22%. The implementation of progressive dropout, enhanced data augmentation, and optimized architecture resulted in a more robust model that better handles the complexities of Pokemon classification across 150 distinct classes. While there remains room for further improvement, the current results demonstrate the potential of deep learning in handling complex image classification tasks, particularly in scenarios involving diverse class distributions and limited training data.

10. REFERENCES

- *7,000 Labeled Pokemon*. (n.d.). Www.kaggle.com. https://www.kaggle.com/datasets/lantian773030/pokemonclassification/data
- Awan, A. A. (2022, November). *A Complete Guide to Data Augmentation*. Www.datacamp.com. https://www.datacamp.com/tutorial/complete-guide-data-augmentation
- GeeksforGeeks. (2019, December 12). *Directory traversal tools in Python*. GeeksforGeeks. https://www.geeksforgeeks.org/directory-traversal-tools-in-python/
- *machine learning - Cross Entropy vs. Sparse Cross Entropy: When to use one over the other*. (n.d.). Cross Validated. https://stats.stackexchange.com/questions/326065/cross-entropy-vs-sparse-cross-entropy-when-to-use-one-over-the-other
- *Papers with Code - EfficientNet Explained*. (n.d.). Paperswithcode.com. https://paperswithcode.com/method/efficientnet
- Prathammodi. (2023, October 16). *Convolutional Neural Networks for Dummies*. Medium. https://medium.com/@prathammodi001/convolutional-neural-networks-for-dummies-a-step-by-step-cnn-tutorial-e68f464d608f
- *Set your Classify confidence score | Zonos Docs*. (n.d.). Zonos. https://zonos.com/docs/supply-chain/classify/features/confidence-score
- Team, K. (n.d.-a). *Keras documentation: ModelCheckpoint*. Keras.io. https://keras.io/api/callbacks/model_checkpoint/
- Team, K. (n.d.-b). *Keras documentation: Whole model saving & loading*. Keras.io. https://keras.io/api/models/model_saving_apis/model_saving_and_loading/
- *What Is a Convolutional Neural Network? | 3 things you need to know*. (n.d.). Www.mathworks.com. https://www.mathworks.com/discovery/convolutional-neural-network.html
- *What is Adam Optimizer?* (2020, October 22). GeeksforGeeks. https://www.geeksforgeeks.org/adam-optimizer/
- *What is Data Stratification?* (n.d.). Study.com. https://study.com/academy/lesson/what-is-data-stratification.html