Project Proposal

Image and Video Processing with Deep Learning

Suchet Samir Sadekar (20221233)

1 Problem Statement

In the Pokémon world, a Pokédex is an electronic encyclopaedia used to record and access information about the various Pokémon species. It uses a camera to recognise a pokémon's species and displays their type, strengths, weaknesses, abilities, and other details about their appearance. My project aims to build a Convolutional Neural Network to identify a pokémon's species from an input image and use a lookup table/database to provide information about its type, strengths, and weaknesses. Due to constraints on time and computing power, only the original 151 pokémon released in the game's first edition will be considered.

2 Input and Output Format

INPUT: 128×128 pixel image.
OUTPUT: A (sorted) list of the top predictions, the number of results displayed will depend on the values of the output probabilities.

3 Pre Processing

Image data for the project was downloaded from Kaggle. Only the 151 Gen-1 pokémon were retained from this database. The data for pokémon types was taken from pokemondb.net. Since the dataset contains only ≈ 40 images per pokémon, image augmentation involving flipping, skewing, shearing, cropping and changing the rotation and brightness of the images was performed using the `Augmentor` module. This produced 500 images per pokémon.
The 75,500 images (151×500) were split into 70% training, 15% validation, and 15% testing sets at random. On loading the image, it is normalized such that every pixel value lies in the range $[0, 1]$.

4 CNN Architecture

The network architecture I have chosen comprises 6 Convolutional Blocks followed by 3 Dense Layers. Each convolutional block passes the image through a `torch.nn.Conv2d()` with a `ReLU()` activation function and a `torch.nn.MaxPool2d()` layer. Each `Conv2d` layer uses a kernel size of size 3×3 pixels with 1-pixel padding on either side. The 6 convolutional blocks reduce the $128 \times 128 \times 3$ tensor into a $1 \times 1 \times 1024$ tensor, which is flattened to a 1D tensor of length 1024. The flattened tensor is then passed through 3 Dense layers with a `torch.nn.ReLU ()` activation function. The 3^{rd} dense layer produces a 1D tensor of length 151 as output. A `torch.nn.SoftMax()` function is applied to the output tensor to convert the values to probabilities.
A cross-entropy loss is used as the loss function as is common in classification problems, and Stochastic Gradient Descent is used as the optimizer. The network has 6,983,127 parameters.

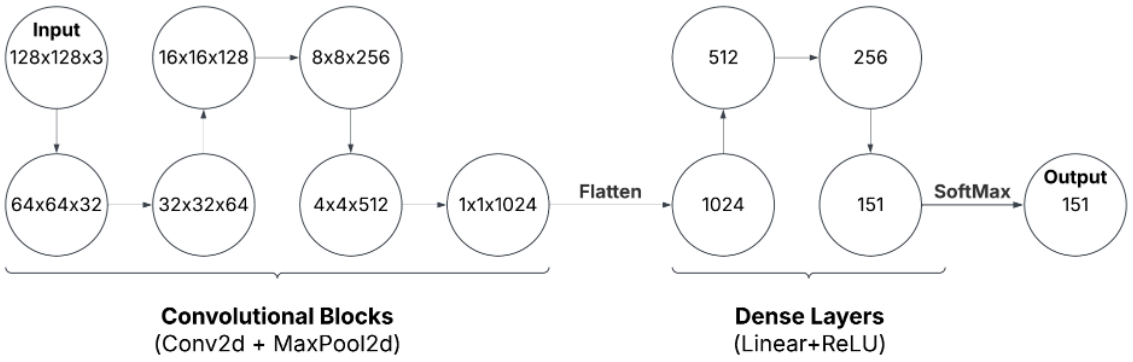

Figure 1: Network Architecture

Since the classification problem involves a large number of classes, the model needs to be at least moderately complex to capture the wide range of features for all 151 classes, although time constraints on training the network warrant that the network cannot be too complex. Hence, I have chosen this network architecture.

5 Post Processing

The vector of probabilities is used to evaluate the identity of the pokémon in the input image. A lookup table is then used to identify its type, strengths, and weaknesses, and the output is displayed appropriately. For high-confidence predictions, only a single prediction will be displayed. For lower confidence predictions, up to 5 pokémon with the highest probability will be displayed, sorted by probability.