Report Tp3_DL

binome : Atek ABDEL JALIL, IGUERNAISSI RIAD

Training a CNN From Scratch :

This report explores the development and training of a Convolutional Neural Network (CNN) designed to conquer the CIFAR-10 dataset. This dataset holds 60,000 images belonging to 10 distinct classes, making it a popular benchmark for testing image recognition algorithms. Our custom-built CNN tackles the challenge of accurately classifying these images, pushing the boundaries of image recognition.

1. Data Acquisition and Preprocessing:

- We obtained the CIFAR-10 dataset, containing 60,000 images categorized into 10 classes (e.g., airplanes, cars, dogs).
- To ensure the model efficiently learns from the data, we normalized the pixel values of each image in the training and testing sets. This scales all pixel intensities to a range of 0 to 1.
- To improve the model's robustness and ability to generalize to unseen data, we implemented data augmentation using ImageDataGenerator. This technique introduces random transformations (rotations, shifts, flips) to the training images, essentially creating new variations without altering the original dataset.

2. Convolutional Neural Network Architecture:

Our image classification task is addressed through a custom-designed Convolutional Neural Network (CNN) architecture implemented using Keras. This architecture leverages a series of convolutional and pooling layers to extract increasingly complex features from the input images, culminating in a final classification stage.

- **Input Layer:** The network commences with an input layer that accepts tensors of dimension (32, 32, 3). This corresponds to the width, height, and RGB color channels of each image within the CIFAR-10 dataset.
-
- **Feature Extraction Layers:**

- The core of the CNN consists of stacked convolutional layers tasked with feature extraction. These layers employ learnable filters (kernels) that scan the input image, identifying and activating upon detection of specific patterns.
  - Two sets of convolutional layers are implemented, each utilizing 16 filters of size 3x3 and employing the rectified linear unit (ReLU) activation function. A valid padding scheme is adopted to maintain the spatial dimensions of the feature maps post-convolution.
  - Following each convolutional layer set, a max-pooling layer with a kernel size of 2x2 is incorporated. This layer performs downsampling, reducing the feature map dimensionality while preserving critical information.
  - Subsequently, a second set of convolutional layers is introduced, this time featuring 32 filters of size 3x3 and employing ReLU activation. Valid padding is once again utilized.
  - Another max-pooling layer with a kernel size of 2x2 is included for further dimensionality reduction.

- **Classification Layer:**

  - After feature extraction, the network transitions to the classification stage:
    - A flatten layer transforms the pooled feature maps into a one-dimensional vector, suitable for feeding into the final dense layer.
    - A terminal dense layer with 10 output neurons and softmax activation is employed to classify the input image into one of the 10 classes present in CIFAR-10. The softmax activation guarantees that the output probabilities for each class sum to 1, providing a well-defined probability distribution over the class labels.

3. Model Training:

- To train the CNN effectively, we employed the Adam optimizer, a popular optimization algorithm known for its efficiency in handling various neural network architectures. It was utilized with its default settings.
- A batch size of 64 was chosen for training. This represents the number of samples processed by the model during each update step. We opted for this value to strike a balance between computational efficiency and gradient estimation accuracy.
- Categorical cross-entropy was selected as the loss function. This function is well-suited for multi-class classification problems like CIFAR-10, measuring the discrepancy between the predicted class probabilities and the true labels.
- Accuracy was monitored as the primary metric to gauge the model's performance during training.

- To gain insights into the training process and prevent overfitting, we tracked both the training and validation loss and accuracy after each epoch. An epoch signifies one complete pass through the entire training dataset.

Following these steps, the model underwent training for a total of 5 epochs. The history object returned by the model.fit() function provided access to the training and validation metrics throughout the process.

Results :

**Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy**
1	1.6064	0.4185	1.3885	0.5012
2	1.2909	0.5449	1.1945	0.5753
3	1.1478	0.5961	1.0739	0.6243
4	1.0472	0.6328	1.0088	0.6452
5	0.9825	0.6567	0.9902	0.6552

4.Improving CNN Performance:

1-More filters :

Our initial CNN achieved a validation accuracy of around 65%. To push for a target of 70% accuracy on the test set with only 5 epochs, we opted to explore the impact of increasing the number of filters within the convolutional layers**. from 16 to 32 and from 32 to 64 .**

A new CNN architecture was designed, incorporating a higher filter count in both sets of convolutional layers. This modification aimed to enhance the model's feature extraction

capabilities by providing a larger number of filters to learn from the input images.

result

**Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy**
1	1.5134	0.4521	1.2413	0.5521
2	1.1200	0.6070	1.0317	0.6392
3	0.9502	0.6683	0.9460	0.6694
4	0.8443	0.7077	0.8731	0.6934
5	0.7733	0.7315	0.8459	0.7071

2-regularization (dropout) :

Enhancing Model with Regularization:

In an effort to further improve the CNN's performance and potentially mitigate overfitting, dropout layers were incorporated into the architecture. Dropout is a regularization technique that randomly drops a certain percentage of neurons during training, preventing the model from overrelying on specific features.

The dropout rate of 0.25 signifies that 25% of the neurons in the layer are randomly deactivated during each training iteration. This helps prevent the model from forming overly specific connections to the training data, encouraging it to learn more generalizable features.

result :

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
1	1.5877	0.4204	1.3129	0.5216
2	1.2189	0.5691	1.2517	0.5643
3	1.0678	0.6247	1.0024	0.6550
4	0.9649	0.6616	0.8986	0.6858
5	0.8941	0.6882	0.8391	0.7108

Comparative table :

Feature	Initial CNN (5 Epochs)	CNN with More Filters (5 Epochs)	CNN with Dropout (5 Epochs)
Validation Accuracy	~65% (Estimated)	69.34% (Epoch 4)	71.08% (Epoch 5)
Training Behavior	Likely decreasing loss & increasing accuracy (5 epochs)	Observed decreasing loss & increasing accuracy (5 epochs)	Observed decreasing loss & increasing accuracy (5 epochs)
Overfitting Potential	Moderate (based on estimated validation accuracy)	Potentially lower (due to higher validation accuracy with more filters)	Potentially even lower (highest validation accuracy)
Improvement	Baseline	Achieved higher validation accuracy (and potentially less overfitting) within 5 epochs	Achieved the highest validation accuracy (potentially due to reduced overfitting) within 5 epochs