Norine Ndona NDOUDI

Asssignment 7 Report

This report will analyze three CNN models developed for the task of classifying images from the CIFAR-100 dataset. After trying different combinations of activation functions, optimizers, hyperparameters, and model architectures, we have selected 3 models which were the most effective within those we tried.

These are the models chosen:

Model 1:

- Architecture:
  - A Convolutional Neural Network with two convolutional layers: both has a 3x3 kernel, one with 32 filters, the other with 64 filters.
  - A max-pooling layers with a 2*2 pool size after each convolutional layer
  - A flattening layer
  - A denser layer with 128 units
  - The activation function used is the RELu for the hidden layers.
  - An output dense layer with 100 units (number of different classes) with softmax activation function.
- We used the Adam optimizer.
- We used the Categorical cross-entropy loss function.
- Hyperparameters:
  - Learning rate: 0.001
  - Number of epochs: 10
  - No minibatch
- Test accuracy: 32.3%
- Total number of parameters: 327332

Model 2:

- Architecture:
  - A Convolutional Neural Network with three convolutional layers: all has a 3x3 kernel, one with 32 filters, the other with 64 filters and the last one with 128 filters.
  - A max-pooling layers with a 2*2 pool size after each convolutional layer
  - A flattening layer
  - A fully connected dense layer with 512 units
  - Dropout layer with a rate of 0.5.
  - The activation function used is the RELu for the hidden layers.
  - An output dense layer with 100 units (number of different classes) with softmax activation function.
- We used the Adam optimizer.
- We used the Categorical cross-entropy loss function.

- Hyperparameters:
  - Learning rate: 0.001
  - Number of epochs: 40
  - Batch size: 256
- Test accuracy: 33.8 %
- Total number of parameters: 407204

Model 3:

- Architecture:
  - A Convolutional Neural Network with two convolutional layers: both has a 3x3 kernel, one with 64 filters, the other with 128 filters.
  - A max-pooling layers with a 2*2 pool size after each convolutional layer
  - A flattening layer
  - A denser layer with 128 units
  - The activation function used is the tanh for the hidden layers.
  - An output dense layer with 100 units (number of different classes) with softmax activation function.
- We used the Adam optimizer with beta values (beta_1 = 0.9, beta_2 = 0.999).
- We used the Categorical cross-entropy loss function.
- Hyperparameters:
  - Learning rate: 0.001
  - Number of epochs: 20
  - Batch size: 256
- Test accuracy: 32.1 %
- Total number of parameters: 678500

To evaluate the effectiveness of our models in the context of image classification on the CIFAR-100 dataset, we compared the performance of our models against top-performing models from the CIFAR-100 image classification leaderboard which didn't use extra training data. We collected benchmarking results from various sources to put our models' performance into perspective. Here are some key benchmarking results:

- Stochastic Pooling: Percentage correct 57.5%
- NiN (Network In Network): Percentage correct 64.3%
- DSN (Deeply-Supervised Nets): Percentage correct 65.4%
- HD-CNN (Hierarchical Deep Convolutional Neural Network for Large Scale Visual Recognition): Percentage correct 67.4
- Spectral Representations for Convolutional Neural Networks: Percentage correct 68.4
- Dspike (ResNet-18): Percentage correct 74.24%
- WRN-28-8 (SAMix+DM): Percentage correct 85.59%
- Dynamics 1 (Particle Swarm Optimization): Percentage correct 87.48%
- Astroformer: Percentage correct 93.36%

As we can see, the range of the test accuracy of my models is [0.321; 0.338] compared to the performance of other models on the CIFAR-100 using other type of model, hyperparameters, regularization and so on…

I think maybe modify some hyperparameters can help to improve the performance of my models, maybe try the SGD optimizer as used in some of the top-performing models on the leaderboard.

In fact, for the 3rd model, I should reduce the number of parameters because the model is overfitting. I can also add a regularization techniques like dropout or L2 regularization to mitigate overfitting.
For the two other models, I should increase the number of epochs (if computational resources permit because the runtime is very long just for 10 epochs) but also maybe increase the model's capacity by adding more units. At the end, redoing a trial-and-error process.