Deep Learning KU (708.220) WS23

Assignment 3: Modern Deep Architectures

We will use TensorFlow and Keras to train a CNN for the grayscale CIFAR-20 image classification dataset. This dataset is an alternative representation of the CIFAR-100 dataset[1], where the 100 classes are grouped into 20 superclasses. It originally consists of 50,000 training and 10,000 test images of size 32x32 in RGB (see Fig. 1).

We will begin by loading the dataset with the 20 *coarse* superclass labels, and converting all images to grayscale via averaging the three RGB channel pixel values. You can load the dataset and convert into grayscale by executing:

```
import tensorflow.keras.datasets as tfd
train_data, test_data = tfd.cifar100.load_data(label_mode="coarse")
(x_train, y_train), (x_test, y_test) = train_data, test_data
x_train, x_test = np.mean(x_train, axis=3), np.mean(x_test, axis=3)
```

Here, class labels are indicated with integers from 0–19, which should be converted to a one-out-of-K encoding (K is the number of classes) to train the network.

Your task will be to construct a CNN architecture that both achieves high performance on the original grayscale CIFAR-20 test set examples, and also demonstrates good generalization performance on various image perturbations. Provided file `cifar20_perturb_test.pkl` contains a *perturbed test set* for these additional evaluations (see Fig. 2). You can load this file and convert to grayscale via:

```
import pickle
dict = pickle.load(open('cifar20_perturb_test.pkl', 'rb'))
x_perturb, y_perturb = dict['x_perturb'], dict['y_perturb']
x_perturb = np.mean(x_perturb, axis=3)
```

We expect a convolutional neural network model that can obtain high prediction performance on both `x_test` and `x_perturb`. Towards achieving this goal, for all the tasks below, create the appropriate code and discuss your experimentation and reasoning process, findings and choices in your report.

Task details:

a) (2 pts) : Load and prepare your data and labels. Familiarize yourself with the dataset and problem. Construct a stratified validation set consisting of samples from the training data, which will be used during the model selection process. You will use the test set only for final evaluations.

[1] `https://keras.io/api/datasets/cifar100/`

b) (8 pts) : Design and train your customized CNN for this multi-class classification task, that achieves good performance on the grayscale CIFAR-20 test set. Explain your choices for the output layer and the error function that you will use. During training with mini-batches, use early stopping based on the validation set to avoid overfitting. Test different architectures with varying numbers of convolutional layers and number of kernels (i.e., change the architecture in depth and width). Compare training and validation set errors and accuracies of (at least five) different architectures and report in a table.

c) (5 pts) : Investigate and compare different layer weight regularizers and dropout for regularization. You should use the validation set to find the appropriate regularization hyper-parameters (e.g., l_2-regularization weight, or a good dropout rate). Provide a table where training and validation set errors and accuracies of various regularization settings are compared.

d) (3 pts) : Summarize your final model architecture in detail (i.e., convolutional and/or pooling kernel sizes and specifications, input and output dimensionalities per layer, activation functions). Report your training specifications regarding the loss function, optimization scheme and learning rate, batch size and number of training epochs, and the amount of trainable parameters in your network. Provide a plot where the evolution of the training and validation error during training is shown throughout iterations. Perform a final training with the whole training set. Report the final test accuracy and a confusion matrix of your test set predictions.

e) (7 pts) : Now, evaluate your CNN from part (d), on the *perturbed test set and report the accuracy.* Then, try and discuss different approaches (e.g., regularization, batch normalization, data augmentation[2], etc.) to achieve better generalization of your model against such image perturbations. Demonstrate in a single table the trade-offs between the training set, test set, and *perturbed test set* accuracies under various modifications to your model. Present your final architecture and optimization scheme clearly, and report the final training set, test set, and *perturbed test set* accuracies. You are **not** allowed to use samples from the *perturbed test set* to fine-tune your model weights.

Total: 25 points

Present your results clearly and structured. Submit your commented code (a single .py file) and a report (.pdf) at TeachCenter. Do **not** provide one zip file with code and PDF, but submit them separately. Your code should be directly executable (assuming that the provided files are present at the working directory).

[2] `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`

Assignment details:

- *Assignment issued:* November 22nd, 2023, 08:00
- *Deadline:* December 20th, 2023, 08:00
- *Solution submission:* Upload to TeachCenter a PDF (report) and a single .py (code) file, separately as explained above (i.e., **not** in a zip file).
- *Rules:* **Groups of up to two students are allowed for this task.** In the report, indicate your group partner on top of the first page (write "Group partner: ⟨First name⟩, ⟨Last Name⟩, ⟨Matrikelnumber⟩"). It is sufficient if only one of the group members submits the group's solutions. Copying of solutions or reports from other students is strictly forbidden.

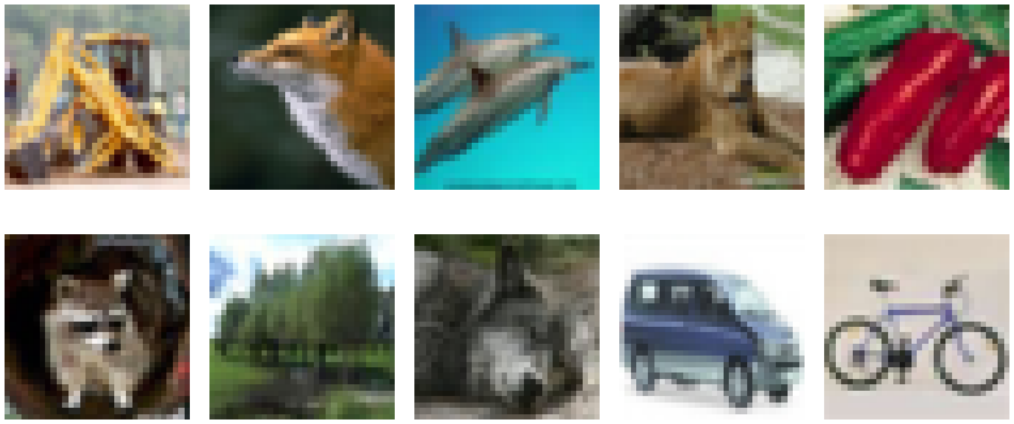

Figure 1: Example images from the original dataset.

Figure 2: Example *perturbed test set* images.