Einführung in die Technische Informatik
Design and Implementation of Deep Neural Networks

Salim Ullah

Chair of Processor Design
Fakultät Informatik Technische Universität Dresden

Wintersemester 2019/20
Table of Contents

1. ANN Design using Keras
2. Exercises
3. CNN Design using Keras
4. Exercises
Design a light-weight DNN for Image Classification

Different benchmark training, validation and testing datasets
- MNIST handwritten digit dataset
- MNIST fashion dataset
- CIFAR-10 and CIFAR-100 dataset
- ImageNet dataset

Some datasets are already provided by TensorFlow

We will be looking at MNIST dataset in this example
If it doesn’t work on MNIST, it won’t work at all
Well, if it does work on MNIST, it may still fail on others
A training set of 60,000 examples, and a test set of 10,000 examples
Original dataset contained ten handwritten digits
MNIST fashion dataset contains items from 10 different classes
28 × 28 grayscale images
Can be accessed from TensorFlow directly
MNIST Dataset from TensorFlow

- Loading the dataset returns four NumPy arrays
  - The `train_images` and `train_labels` arrays are the training set—the data the model uses to learn
  - The model is tested against the test set, the `test_images`, and `test_labels` arrays.
- The images pixel values range from 0 to 255
- The labels are an array of integers, ranging from 0 to 9
Data classes

<table>
<thead>
<tr>
<th>Label</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>T-shirt/top</td>
</tr>
<tr>
<td>1</td>
<td>Trouser</td>
</tr>
<tr>
<td>2</td>
<td>Pullover</td>
</tr>
<tr>
<td>3</td>
<td>Dress</td>
</tr>
<tr>
<td>4</td>
<td>Coat</td>
</tr>
<tr>
<td>5</td>
<td>Sandal</td>
</tr>
<tr>
<td>6</td>
<td>Shirt</td>
</tr>
<tr>
<td>7</td>
<td>Sneaker</td>
</tr>
<tr>
<td>8</td>
<td>Bag</td>
</tr>
<tr>
<td>9</td>
<td>Ankle boot</td>
</tr>
</tbody>
</table>
Steps for building and training the model

- Download the training and testing datasets
- Scale the data to the range 0 to 1 for better training
- Build a sequential model for an ANN using Keras
  - Input layer: `tf.keras.layers.Flatten`, transforms the format of the images from a two-dimensional array (of \(28 \times 28\) pixels) to a one-dimensional array (of \(28 \times 28 = 784\) pixels)
  - Input layer has no parameters to learn; it only reformats the data.
  - After that, a sequence of two `tf.keras.layers.Dense` layers.
  - The first `Dense` layer has 128 nodes (or neurons).
  - The second (and last) layer is a 10-node softmax layer.
Steps for building and training the model

- Compile the model
  - Define the Loss function
  - Define the optimizer
  - Define a metric monitoring the training
- Train the model using `model.fit` function
- Evaluate the model using `model.evaluate` function
- Individual predictions about test images can be made using `model.predict` function
Exercises

- Look at the template of the `keras.layers.Dense`. It uses a default initializer for weights and biases. Try some other initializer.
- Change the number of layers and number of nodes in each layer and observe the change in classification accuracy.
- Plot a graph showing the classification accuracy with different number of layers and different number of nodes in each layer. Basically we will be looking at the overfitting phenomenon.
- Try different optimizer and loss functions and observe the change in classification accuracy.
- Try training with a subset of the training images. How does it affect the classification accuracy?
- Take some random image (may be a cat). Crop it to $28 \times 28$ and then convert to grayscale. Provide this image to the trained network. What does it tell you about the cat image?
CNN Design using Keras

- CNN: An ANN with Convolution layers
- Design a CNN for CIFAR-10 dataset
  - 60,000 color images in 10 classes, with 6,000 images in each class
  - Size of image: 32 \times 32
  - The dataset is divided into 50,000 training images and 10,000 testing images
## CIFAR-10 Classes

<table>
<thead>
<tr>
<th>Label</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>airplane</td>
</tr>
<tr>
<td>1</td>
<td>automobile</td>
</tr>
<tr>
<td>2</td>
<td>bird</td>
</tr>
<tr>
<td>3</td>
<td>cat</td>
</tr>
<tr>
<td>4</td>
<td>deer</td>
</tr>
<tr>
<td>5</td>
<td>dog</td>
</tr>
<tr>
<td>6</td>
<td>frog</td>
</tr>
<tr>
<td>7</td>
<td>horse</td>
</tr>
<tr>
<td>8</td>
<td>ship</td>
</tr>
<tr>
<td>9</td>
<td>truck</td>
</tr>
</tbody>
</table>
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', activation=None, use_bias=True, kernel_initializer='glorot_uniform')

- A CNN takes tensors of shape (image_height, image_width, color_channels)
- The output of every Conv2D and MaxPooling2D layer is a 3D tensor of shape (height, width, channels)
- The width and height dimensions tend to shrink as you go deeper in the network.
- The number of output channels for each Conv2D layer is controlled by the first argument (e.g., 32 or 64).
Exercises

- Try convolution layers with different numbers of neurons and filter sizes.
- Design and train a network for the classification of CIFAR-100 dataset.
- Try to store and plot the weights and biases of the trained network.
- Try to store a trained network and then reload it again.