Universidad Complutense de Madrid x NTIC
Manuel Sanchez Mc Sweeney
Deep Learning

Project Report: Deep Learning I & II Examination

1. Introduction

This document provides a summary of the practical examination for the "Deep Learning I (Fundamentals)" and "Deep Learning II (Image Processing)" modules. The project was developed in a Jupyter Notebook environment and was designed to be executed on either a local machine or Google Colab to leverage GPU capabilities for training neural networks. The examination is structured into two main activities, each worth 5 points, covering fundamental concepts of dense neural networks and more advanced applications of convolutional neural networks. The final submission required all code and text cells to be fully executed.

2. Activity 1: Dense Networks for Regression

Objective: The first part of the exam focused on building and refining a dense neural network to solve a regression problem. The goal was to predict the quality of wine using the public "Wine Quality" dataset from the UCI Machine Learning Repository.

Dataset and Preprocessing:

- The dataset combines data for both red and white wines.
- The features include physicochemical properties like 'fixed acidity', 'citric acid', and 'alcohol'.
- The target variable is 'quality', which is a continuous value, making this a regression task.
- A key requirement was to correctly normalize the input features before feeding them into the network, for which up to 0.25 points were awarded.

Tasks and Models:

1. **Baseline Model:** The first task was to create a baseline sequential model using TensorFlow Keras. The architecture required at least four hidden layers, each with more than 60 neurons, and an appropriate output layer for regression. The model was to be compiled with a suitable loss function for this type of problem.
2. **Regularization:** The second task involved taking the baseline model and incorporating at least two different regularization techniques (e.g., Dropout, L2 regularization) to combat overfitting. A key success criterion was achieving a lower test loss than the baseline model.
3. **Early Stopping:** The final modeling task was to add an `EarlyStopping` callback to the regularized model. This technique helps in preventing overfitting by stopping the training process once the model performance on a validation set ceases to improve.

Theoretical Questions: This activity also included several theoretical questions to test fundamental knowledge:

- Potential alternative activation functions for the output neuron in a regression problem.
- The core calculation performed by a single neuron.
- Which activation functions are generally not recommended for hidden layers.
- Identifying all valid techniques for combating overfitting from a given list.
- Determining the output layer configuration (number of neurons, activation function) and the appropriate loss function for a multi-class classification problem.

---

3. Activity 2: Convolutional Networks for Image Classification

Objective: The second activity shifted focus to computer vision, requiring the construction of a Convolutional Neural Network (CNN) for image classification. The goal was to achieve a test accuracy of over 68% on the CIFAR-10 dataset.

Dataset:

- The CIFAR-10 dataset was used, which contains 60,000 32x32 color images distributed across 10 distinct classes (e.g., 'airplane', 'cat', 'dog', 'ship').
- The data was pre-loaded using **`tf.keras.datasets.cifar10.load_data()`.**

Tasks and Models:

1. **Functional API Model:** The main task was to build a CNN using the Keras Functional API. The network architecture had to include:
    - An input layer appropriate for the shape of the CIFAR-10 images.
    - At least two convolutional layers (**`Conv2D`**) and two pooling layers (**`MaxPooling2D`**).
    - A **`Flatten`** layer to transition from 2D feature maps to a 1D vector.
    - A final **`Dense`** output layer with the correct number of neurons and activation function for 10-class classification.
    - The model was compiled with the 'adam' optimizer and **`SparseCategoricalCrossentropy`** loss function.
2. **Sequential Model Conversion:** A follow-up task required converting the same architecture from the Functional API into a **`tf.keras.models.Sequential`** model, demonstrating versatility with both Keras APIs.

Theoretical Questions: This section concluded with questions focused on CNNs:

- Calculating the number of parameters in the first hidden layer of a dense network if it were used on a 300x300 color image.
- Identifying the primary advantages of CNNs compared to dense networks for image processing tasks.
- A true/false question regarding the effectiveness of CNNs for processing time-series data.