

## Midterm 1 - Deep Learning - Spring 2025

**Essay Question:** Write a one paragraph essay discussing *how graph convolutional networks solve the problem of having a limited amount of labeled data samples.*

**Problem 1:** Consider a learning-by-example problem that trains a model that takes a journal article whose text is one-hot encoded with respect to a dictionary of 1500 words and then outputs an estimate of the probability that the article belongs to a given class (science, sports, entertainment, etc.). Assume the article classes are one-hot encoded and that your model outputs probability estimates for 7 distinct classes.

- Let  $\{(x_k, y_k)\}_{k=1}^N$  denote a finite dataset created by the system's generator and observer. Formally describe the input samples,  $x_k$ , and target samples,  $y_k$ .
- There are several model sets you can choose for this problem. Select one and formally describe it.
- Select a loss function for your problem, write out the equation for the loss function, and explain what the variables are in the equation.

**Problem 2:** Consider a model set,  $\mathcal{H}$  with finite VC dimension of 3 and assume we have trained a model  $h^* \in \mathcal{H}$  that minimizes the empirical risk of a training dataset having 1000 samples. You then use a separate test dataset consisting of 50 samples and find that the optimal model's empirical risk on the test data is 0.1. Determine an upper bound on the optimal model's true risk with a confidence level of  $1 - \frac{2}{e^3}$ .

**Problem 3:** Consider the 1D convolutional model,  $h : \mathbb{R}^3 \rightarrow \mathbb{R}$ , instantiated with the following script

```
from tensorflow import keras
from tensorflow.keras import layers
inputs = keras.Input(shape=(3,1))
outputs = layers.Conv1D(filters = 1, kernel_size = 3, strides = 2,
    activation = "relu", use_bias = False)(inputs)
model = keras.Model(inputs = inputs, outputs = outputs)
model.compile(loss = "MSE")
```

List all nodes for a computation graph of this model along with each node's name, "expression" and node type. Use this list to sketch the computation graph for the model labeling each node with its name and expression, and drawing edges between the appropriate nodes.

**Notebook Problem:** You are to train a model that detects heart arrhythmias from electrocardiogram (ECG) traces measured using a user's smartwatch. An arrhythmia is an abnormal heart rhythm that can be caused by a number of medical conditions. The dataset you will be working with is the [MIT-BIH Arrhythmia database](#). A pre-processed version of this database has been included as a pickle file in the midterm's data directory. The input samples are float64 vectors of length 32 representing the ECG time series. The associated targets are one-hot encoded vectors in which  $[0.0, 1.0]$  denotes a *normal* sample and  $[1.0, 0.0]$  denotes an *arrhythmia* sample.

- **Part 1:** Load the training and test datasets from the pickle file in the `data` directory. The imported dataset is a list of length 4. The first entry in the list is the training inputs, the second list entry is the training targets, the third list entry is the testing inputs, the last list entry is the testing targets. *Determine the number of normal and arrhythmia training samples. Plot one of the ECG traces for a normal and arrhythmia sample.*
- **Part 2:** Instantiate the model architecture in Fig. 1. This model has a convolutional base with two 1D convolutional layers (Conv1D) with `strides` of 2 and `kernel_size` of 3. The first Conv1D layer has 64 filters and the second one has 128 filters. The output of the convolutional base is fed to a Dense layer with 256 nodes, followed by a Dense output layer with 2 nodes. The model's hidden layers use ReLu activation. The output layer uses a softmax activation. Use a dropout layer after the 256 node dense layer to help avoid overfitting. *Evaluate your untrained model's accuracy on the test dataset. Display the confusion matrix for the untrained model on the test dataset.*

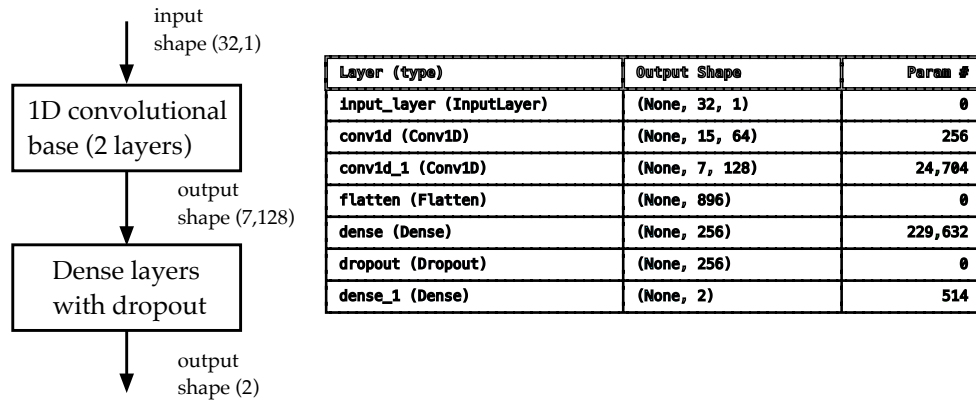


Figure 1: MD1 Notebook Model

- **Part 3:** Fit your model for 20 epochs on the training data assuming a 20% validation split. Use the focal Tversky loss function included in the notebook's utility file, `MD1utils`. List the hyperparameters for the optimizer, regularizer, and batched training data. Adjust the hyperparameters (if necessary) until you achieve a validation accuracy greater than 92%. Plot the training curves, save the best model with respect to validation loss, and determine the best model's loss, accuracy, sensitivity, and specificity on the testing dataset. Display the confusion matrix for the trained model on the test dataset.
- **Part 4:** The original dataset is highly imbalanced. Use Synthetic Minority Oversampling (SMOTE) [1] to rebalance the dataset. If your Python environment doesn't have the `imblearn` library, you will need to install it using

```
pip install imbalanced-learn
```

You can then import SMOTE into your notebook and use the following script to create a SMOTE object that "oversamples" the original data set, (X,y).

```
from imblearn.over_sampling import SMOTE
sm = SMOTE()
X_sm, y_sm = sm.fit_resample(X,y)
```

Determine how many "normal" and "arrhythmia" samples are in the SMOTE dataset you created. Reset your model states/weights and fit it for 20 epochs on the SMOTE dataset assuming a 20% validation split. Use the same set of hyperparameters you used in the preceding part. Plot the training curves, save the best model with respect to validation loss, and determine the best model's loss, accuracy, sensitivity, and specificity on the testing dataset. Comment on the results of your new model versus the one you obtained in the preceding part.

## References

- [1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.