CHEM361: MACHINE LEARNING IN CHEMISTRY
SPRING 2025, UW-MADISON
HOMEWORK 3

Due date: 5pm, March 20, 2025

Instructions: This homework covers "Chapter 5: Neural Networks - Fundamentals and Applications in Chemical Modeling" and "Chapter 6: Deep Neural Networks - Advanced Architectures for Chemical Applications". Use this PDF file as a template to complete your homework. Submit your work on time as a ZIP file to **Canvas**, including:

(1) A single PDF or Word document containing your answers (handwritten or typed).

(2) Your code in a Jupyter Notebook.

You may refer to the provided reference notebooks in chapter 5 and 6, or online resources to help you solve the questions. If needed, you can use Google Colab (`https://colab.research.google.com/`) instead of a local Python environment. For Python package functionality, consult the official documentation for detailed explanations and examples.

There are 4 homework assignments; the 3 highest scores will each contribute 20% to your final grade. ***Late homework submissions will not be accepted.***

1 Predicting Compound Solubility Using Multi-layer Perceptron (MLP) (40 points)

Fetch the Delaney solubility dataset (`https://raw.githubusercontent.com/deepchem/deepchem/master/datasets/delaney-processed.csv`), to complete the tasks in this question. For each chemical compound, use input feature vectors consisting of four molecular properties: **"Molecular Weight", "Number of H-Bond Donors", "Number of Rings", and "Number of Rotatable Bonds"**.

In class, we went through the reference notebook `Reference_Ch5_Part_2_MLP.ipynb`, which demonstrated how to train an MLP for binary classification tasks. In this question, we will train MLP models to predict the "measured log solubility in mols per liter", which is a regression task.

(Hint: A good practice is to include **BatchNorm1d** before all layers to normalize the input data.)

1.1 Please modify the loss function for the regression task. Report which loss function you choose. **(5 points)**

(Hint: Please recall the loss function used in linear regression. You can find commonly-used loss functions in PyTorch here `https://pytorch.org/docs/stable/nn.html#loss-functions`)

1.2 Please modify the **train_one_epoch** and **val_one_epoch** functions to adapt them for the regression task. Briefly discuss the changes you have made. **(5 points)**

1.3 Randomly split the dataset to training and validation sets using a 90:10 ratio. Train an MLP model with three hidden layers containing 10, 5, and 1 (output) neurons, respectively. Please use the stochastic gradient descent (SGD) optimizer with a learning rate of 0.01, a batch size of 128, and train the model for 500 epochs. Please plot the loss values against epochs for both the training and validation process, and report the MSE and R^2 on the validation set. **(15 points)**

1.4 Building on Q1.3, please keep all other hyperparameters unchanged and train the model using different learning rates: $[0.001, 0.1, 1]$. Plot loss values against epochs for both the training and validation process for each learning rate. Explain how the choice of learning rate affects the training process. **(15 points)**

2 Predicting log EC50 of dual-agonist peptides (20 points)

Fetch the training dataset (`https://github.com/xuhuihuang/uwmadisonchem361/blob/main/CNN_training_data.csv`), which has been preprocessed based on data from **Machine learning designs new GCGR/GLP-1R dual agonists with enhances biological potency** (`https://www.nature.com/articles/s41557-024-01532-x`). Please refer to the reference notebook `Reference_Ch6_Part_1_CNN.ipynb`. The neural network architecture is structured as follows:

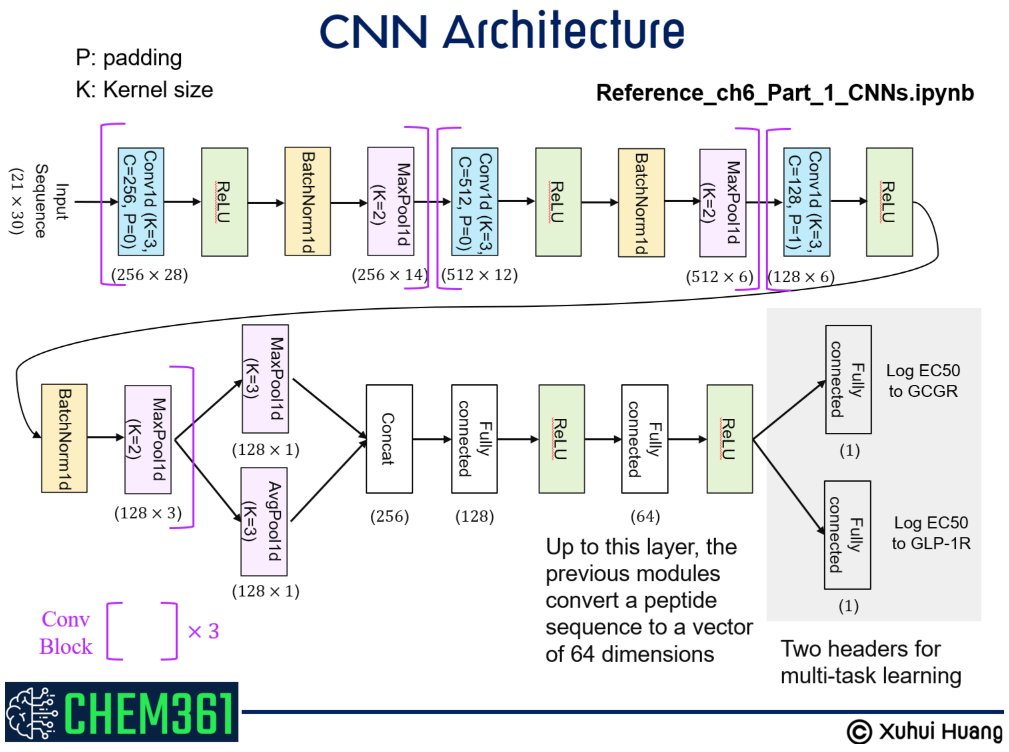

Figure 1: Train Peptide Dataset using CNN

2.1 In this question, we will explore how the number of convolutional blocks affects training results. Modify the architecture (see Figure 2) to include only a single **nn.conv1d** block with the following specifications: kernel size of 25, 128 channels, stride of 1, no padding, and no bias. Please report the total number of parameters in this modified architecture.

(Hint: You may need to modify the model definition, the **forward** function, and possibly the **_calculate_flattened_size** function.) **(10 points)**

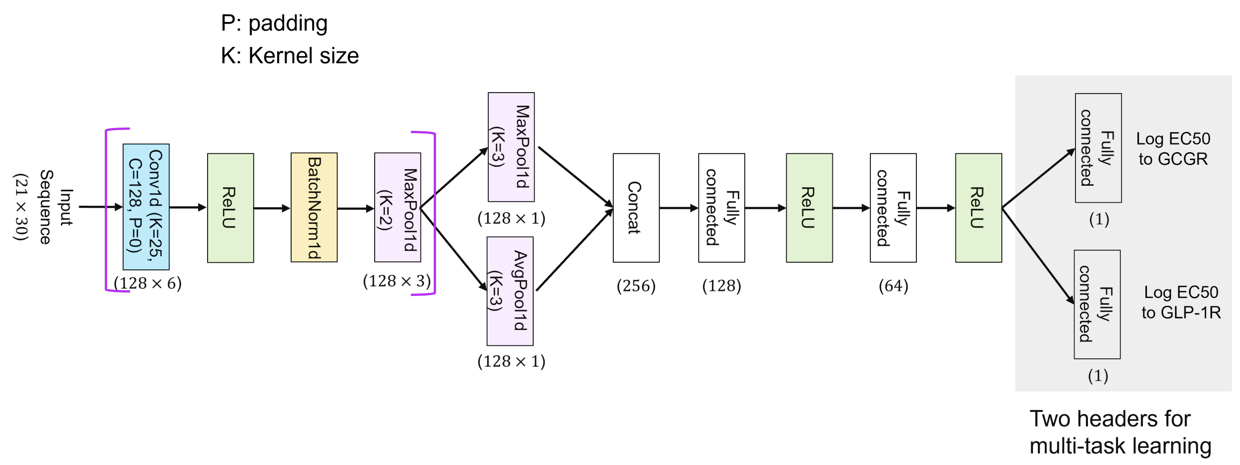

Figure 2: Modified CNN architecture

2.2 Continuing from Q2.1, randomly split the dataset into training and validation sets using a 90:10 ratio. Train the modified CNN model using the AdamW optimizer with a learning rate of 0.001, a batch size of 10, and 1500 epochs. Please report the MSE and R^2 values of the predicted log EC50 of both GPCR receptors (GCGR and GLP-1R) on the validation set separately. **(10 points)**

3 Predicting Compound Solubility Using Recurrent Neural Networks (RNN) (40 points)

Fetch the Delaney solubility dataset (`https://raw.githubusercontent.com/deepchem/deepchem/master/datasets/delaney-processed.csv`), to complete the tasks in this question. Please use SMILES strings to represent each chemical compound and train a recurrent neural network (RNN) to predict the "measured log solubility in mols per litre" value. Please refer to the reference notebook for guidance `Reference_Ch6_Part_2_RNN.ipynb`

3.1 Please use the same tokenizer as in the reference notebook. Report the SMILES string, tokens, and token ids for the compound "2,3',5-PCB". **(10 points)**

3.2 Please compare the mean squared error (MSE) and R^2 of the RNN model with those of the MLP model in Q1.3 and the linear regression model in Homework 2, Q1.1. Rank the predictive power of these models and discuss why certain models may predict more accurately than others. **(10 points)**

3.3 Explore the RNN architecture and experiment with different training techniques. You are encouraged to modify the model's hyperparameters, including but not limited to the RNN cell type, number of layers, hidden dimensions, activation functions, learning rate, weight decay, dropout ratios, optimizer, etc. Find a configuration that outperforms the models in Q1.3 and Q3.2. Report the loss and R^2 of your best model on the validation set. **(20 points)**