 # QUESTION 1: What is the difference between GCNConv and GCN ?

- GCNConv. From the paper "Semi-supervised Classification with Graph Convolutional Networks," the graph convolutional operator. ChebConv. From the paper "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," the chebyshev spectral graph convolutional operator.
- A Graph Convolutional Network (GCN) is a semi-supervised learning approach for graph-structured data. It is based on an efficient convolutional neural network variant that works directly on graphs.

 # QUESTION 2: Adjust the learning rate with 2 smaller and two larger values. Re-run the model training and
 # report your observations. What are the problems with large or small learning rate as we discussed in class ?

- A learning rate is a hyperparameter that determines how much the model changes in response to an error each time its weights are updated. Choosing the correct learning rate is crucial, as a value that is too low may result in a lengthy training process that becomes stuck, while a value that is too high may result in learning too quickly or a training process that is unstable.
- In general, a large learning rate enables the model to learn more quickly at the expense of achieving a suboptimal final set of weights. A slower learning rate may enable the model to learn a more optimal or even globally optimal set of weights, but it may require considerably more time to train.
- We must choose a learning rate that is neither too low nor too high, i.e., we must find the optimal compromise. Adjust the learning rate from high to low during training to slow down as you approach the optimal solution. Alternate between high and low rates of learning to create a hybrid.

 # QUESTION 3: Describe all the steps that take place during an "epoch" as we discussed in class

- A single complete cycle through the training data constitutes an epoch. Typically, this involves numerous steps. If you have

5,000 images and a batch size of 10, an epoch would include the following:

- 5,000 images / (10 images / step) = 500 steps.
- Once all images have been individually processed forward and backward to the network, one epoch has occurred.
- Before beginning training on a neural network model, the number of epochs parameter is set. However, we cannot initially determine the optimal number of epochs for the model. We must determine when the neural network weights have converged based on the neural network architecture and data set.
- It is common practice for neural network models to examine learning curve graphs to determine model convergence. Typically, we plot graphs of loss (or error) versus epoch or accuracy versus epoch. As the number of epochs increases during training, we anticipate a decline in loss and an improvement in accuracy. However, we anticipate both loss and accuracy to level off after a while.
- As is customary, it is suggested that the data set be divided into training and validation sets. This allows us to plot learning curve graphs for various sets. These graphs allow us to determine whether the model has overlearned, underlearned, or if it fits the learning set.
- We anticipate a neural network to converge after a number of training epochs. Depending on the available architecture and data, we can treat the number of training epochs as a hyperparameter.
- As it is a gradient descent-based algorithm, neural network weights are updated iteratively. A single training period is insufficient and leads to underfitness. Given the complexity of real-world problems, training a neural network may require hundreds of epochs.
- As a result, we anticipate that the learning curve graphs will continue to improve until convergence. If we continue to train the model, it will become overfit, and validation errors will increase.

 # QUESTION 4: Where is patience used in the code and how does it affect duration of training of the model ?

 # What do you think would happen if we increase or decrease the patience ? (write your opinion or try with code)

- Patience is used in this code to indicate the number of epochs without further improvement before training is discontinued.
- Each epoch after the dashed line will result in a greater validation error value. Therefore, 10 epochs after the dashed line (since our patience is equal to 10) our model will terminate, as no further improvement has been observed.
- It is possible that after 10 epochs (the value specified for patience), the model begins to improve and the validation error begins to decrease. Therefore, we must exercise extreme caution when adjusting this hyperparameter.

 # QUESTION 5: Only for this question use the TUDataset instead of the GraphSage
 # (for all other questions use GraphSage). For the TUDataset, print out the number
# of graphs, nodes, features, node labels, classification labels and edge features.

```python
# THIS CODE WILL BE USED FOR ANSWERING QUESTION 5 ABOVE

print(len(dataset.graphs))

graph=dataset.graphs[0]

print("\n---- Number of nodes ----")
print(graph.n_nodes)

print("\n---- Nodes features ----")
print(len(graph.x))
print(graph.x[0])
print(len(graph.x[0]))

print("\n---- Node labels ----")
print(len(graph.y))
print(graph.y[0])
print(len(graph.y[0]))

print("\n---- Edge features ----")
print(graph.e)

print("\n---- Number of classification labels in the dataset ----")
print(dataset.n_labels)

print("\n---- Adjancency matrix ----")
print(graph.a)
```

```
1113

---- Number of nodes ----
42

---- Nodes features ----
42
[23.  1.  0.  0.]
4

---- Node labels ----
2
1.0
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-76dcc8fa65a7> in <module>()
     16 print(len(graph.y))
     17 print(graph.y[0])
---> 18 print(len(graph.y[0]))
     19
     20 print("\n---- Edge features ----")
```

 # QUESTION 6 : Explain briefly how the masks work to split the dataset as we explained in class.
 # What are the approximate percentages of the splits here ?

- Assign masks to variables. The size of the mask is the number of nodes. It's a filter going over nodes and making some nodes as part of the training, validation, and test. The long array has true and false masks on them. 21%tr/10%va/69%te

 # QUESTION 7: What are the 3 different stages in building our machine learning model ?
What is the difference between compilation and fitting ?
What are the two model definition approaches, and which is the one spektral offers ?
(mention in lecture, also see Tensorflow.org links included in the Collab posted on the May 3rd lecture)
This is a simple three-layer graph. Begin by creating an input node using the functional API to build this model. By calling a layer on this input object, you can add a new node to the layer graph: The "layer call" action is equivalent to drawing an arrow from "inputs" to this newly created layer. You're "passing" the inputs to the dense layer, and the result is x. You can now create a Model by specifying its inputs and outputs in the layer graph: The functional model is used in this approach.
instead of the Spektral pre-defined GCN, I use my custom-defined (functional) model. Finally, I compile my model and am ready to train it. Because Spektral is designed as an extension of Keras, any Spektral layer can be plugged into a Keras Model without modification.
Because GNN layers typically require two or more inputs, we only need to use the functional API (so no Sequential models for now).
Compilation refers to the process by which a computer converts a high-level programming language into a machine language that the computer understands. A compiler is a software that performs this conversion. High-level lingua franca Programming terms, machine language Fitting is an automatic process that ensures your machine learning models have the individual parameters that are best suited to solving your specific real-world business problem with high accuracy.

 # QUESTION 8: Run this functional versus the sequencial model above.
 # Do you observe any difference in the metrics in Tensorboard ?

- The difference between sequential and functional Keras API: For most problems, the sequential API allows you to build models layer by layer. It is limited in that it does not allow you to create models with multiple inputs or outputs or that share layers. There is a loss of accuracy so you can increase the number of layers, neurons or epochs.

 # QUESTION 9 : Run this with and without sample_weights. Do you observe a difference in the metrics, improving, worse, the same ?

Improving.

With

```
Epoch 1/200
1/1 [==============================] - 0s 417ms/step - loss: 10074400.0000 - acc: 6.3220e-04 - val_loss: 10075362.0000 - val_acc: 1.2293e-04
Epoch 2/200
1/1 [==============================] - 0s 139ms/step - loss: 10075362.0000 - acc: 1.2293e-04 - val_loss: 10077174.0000 - val_acc: 1.2293e-04
Epoch 3/200
1/1 [==============================] - 0s 140ms/step - loss: 10077174.0000 - acc: 1.2293e-04 - val_loss: 10079842.0000 - val_acc: 1.0537e-04
Epoch 4/200
1/1 [==============================] - 0s 132ms/step - loss: 10079842.0000 - acc: 1.0537e-04 - val_loss: 10083421.0000 - val_acc: 1.0537e-04
Epoch 5/200
1/1 [==============================] - 0s 138ms/step - loss: 10083421.0000 - acc: 1.0537e-04 - val_loss: 10087977.0000 - val_acc: 1.0537e-04
Epoch 6/200
1/1 [==============================] - 0s 135ms/step - loss: 10087977.0000 - acc: 1.0537e-04 - val_loss: 10093586.0000 - val_acc: 1.0537e-04
Epoch 7/200
1/1 [==============================] - 0s 130ms/step - loss: 10093586.0000 - acc: 1.0537e-04 - val_loss: 10100324.0000 - val_acc: 1.0537e-04
Epoch 8/200
1/1 [==============================] - 0s 134ms/step - loss: 10100324.0000 - acc: 1.0537e-04 - val_loss: 10108264.0000 - val_acc: 1.0537e-04
Epoch 9/200
1/1 [==============================] - 0s 138ms/step - loss: 10108264.0000 - acc: 1.0537e-04 - val_loss: 10117476.0000 - val_acc: 1.0537e-04
Epoch 10/200
1/1 [==============================] - 0s 133ms/step - loss: 10117476.0000 - acc: 1.0537e-04 - val_loss: 10128023.0000 - val_acc: 1.0537e-04
Epoch 11/200
1/1 [==============================] - 0s 137ms/step - loss: 10128023.0000 - acc: 1.0537e-04 - val_loss: 10139965.0000 - val_acc: 1.0537e-04
<keras.callbacks.History at 0x7f6ff8b992d0>
```

With

```
Epoch 1/200
1/1 [==============================] - 0s 143ms/step - loss: 8004109.0000 - acc: 1.1134e-04 - val_loss: 1112697.6250 - val_acc: 1.5352e-04
Epoch 2/200
1/1 [==============================] - 0s 135ms/step - loss: 8005730.0000 - acc: 1.1134e-04 - val_loss: 1113023.2500 - val_acc: 1.5352e-04
Epoch 3/200
1/1 [==============================] - 0s 128ms/step - loss: 8008116.0000 - acc: 8.9075e-05 - val_loss: 1113446.5000 - val_acc: 1.5352e-04
Epoch 4/200
1/1 [==============================] - 0s 136ms/step - loss: 8011226.5000 - acc: 8.9075e-05 - val_loss: 1113967.0000 - val_acc: 1.5352e-04
Epoch 5/200
1/1 [==============================] - 0s 129ms/step - loss: 8015058.5000 - acc: 8.9075e-05 - val_loss: 1114587.7500 - val_acc: 1.5352e-04
Epoch 6/200
1/1 [==============================] - 0s 132ms/step - loss: 8019636.0000 - acc: 8.9075e-05 - val_loss: 1115315.1250 - val_acc: 1.5352e-04
Epoch 7/200
1/1 [==============================] - 0s 143ms/step - loss: 8025004.5000 - acc: 8.9075e-05 - val_loss: 1116156.6250 - val_acc: 1.5352e-04
Epoch 8/200
1/1 [==============================] - 0s 129ms/step - loss: 8031217.5000 - acc: 8.9075e-05 - val_loss: 1117120.8750 - val_acc: 1.5352e-04
Epoch 9/200
1/1 [==============================] - 0s 135ms/step - loss: 8038343.5000 - acc: 8.9075e-05 - val_loss: 1118217.1250 - val_acc: 1.5352e-04
Epoch 10/200
1/1 [==============================] - 0s 137ms/step - loss: 8046450.0000 - acc: 8.9075e-05 - val_loss: 1119454.3750 - val_acc: 1.5352e-04
Epoch 11/200
1/1 [==============================] - 0s 136ms/step - loss: 8055604.0000 - acc: 8.9075e-05 - val_loss: 1120841.7500 - val_acc: 1.5352e-04
<keras.callbacks.History at 0x7f6ff82041d0>
```

 # QUESTION 10 : Describe what each metric represents, and your observations for each metric changing over the advancing epochs (for that you will need to observe the Tensorboard graphs)

```
Evaluating model.
1/1 [==============================] - 0s 63ms/step - loss: 958981.2500 -
acc: 1.8103e-04
Done.
Test loss: 958981.25
Test accuracy: 0.00018102824105881155
```

Essentially the accuracy of the test is 0 which is terrible. We just give the steps per epoch since we do not do batches. It will return the metrics. It shows tge loss and accuracy. The loss function is row numbers depending on the loss it might be higher or lowers. The testing accuracy is horrible. The loss rate remains the same for the validation and training. But the accuracy drops almost reaching essentially 0.