Assignment 3

The assignment consists of 2 parts, giving up to 5 and 10 points. So, the maximum points for this assignment is 15.

I remind you that the assignment must be done alone. Of course, you may consult with each other but the actual work must be done alone.

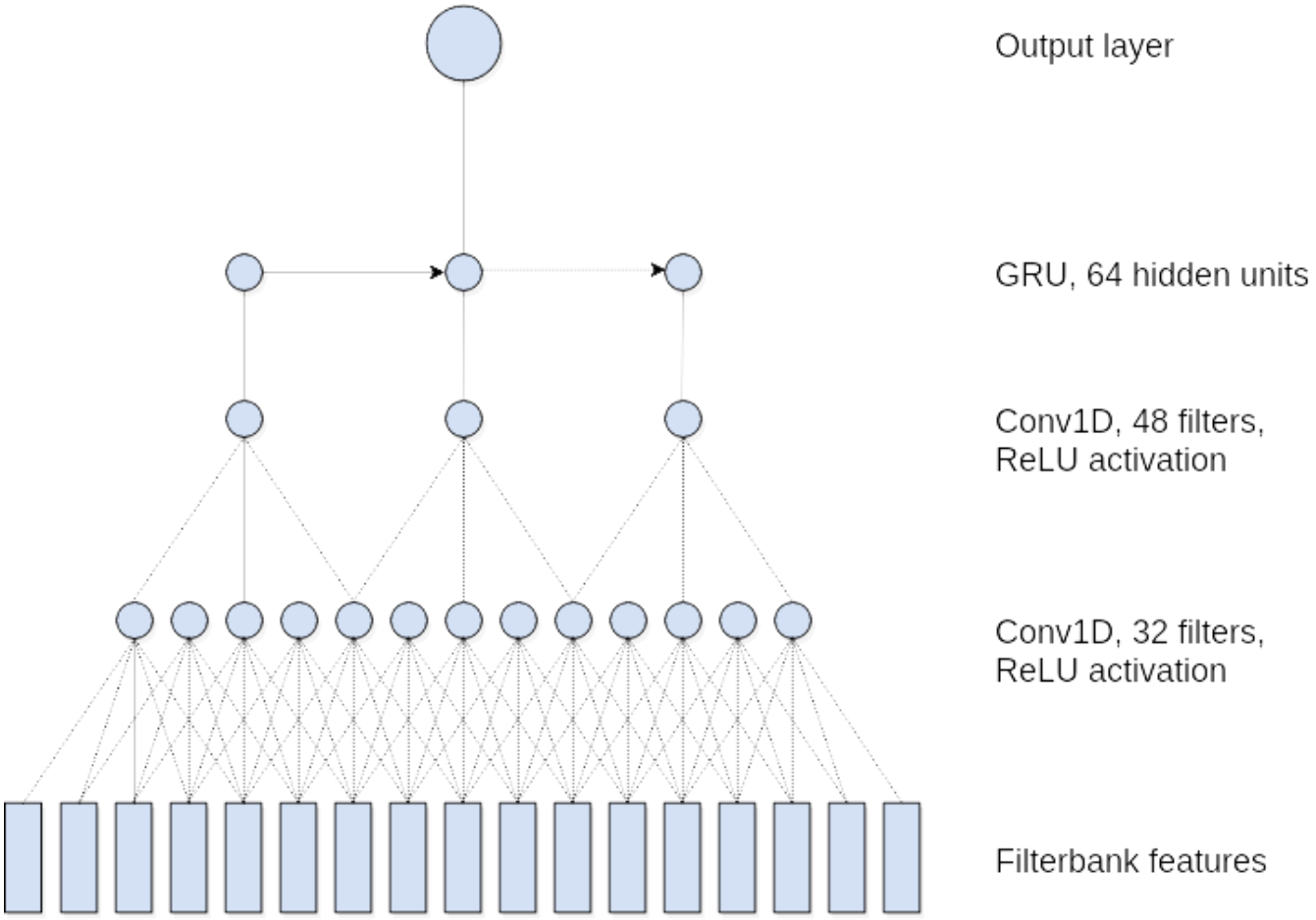
Submit your answers to the question via Moodle in plain text.

All the data loading functions and templates for the models that you have to implement are defined in this Jupyter Notebook:
http://nbviewer.jupyter.org/urls/dl.dropbox.com/s/pxc6xrrzt66faj5/Assignment_3.ipynb

Direct link to the ipynb file:
https://dl.dropboxusercontent.com/s/pxc6xrrzt66faj5/Assignment_3.ipynb (you should be able to directly import this into Google Colab, and then do 'Run All')

Part 1

On May 4 lab we learned how to do speech activity detection. Your task is to implement a model that combines convolutional and recurrent layers, as shown below. Please be accurate: pay attention to how many underlying features the layers takes as input, how many filters or hidden units are used at each layer, pay attention to whether any outputs of the underlying layers are skipped when constructing the inputs for the next layer. You might have to play with the optional arguments of the Conv1D class, documented here:
https://pytorch.org/docs/master/nn.html#torch.nn.Conv1d



Copy/paste the class definition of the model (only). Please use a preformatted code block when pasting the code. Something like:

```
class SadCnnGru(nn.Module):

  def __init__(self, feature_dim):
    ...
```

Points: 5

Part 2

The last part of this assignment is a bit different. Your task is to find a neural network architecture that works best on this speech activity detection task.

Some ideas to explore (they might or might not result in better accuracy):

- Batch normalization can be used not only at input, but also between hidden layers
- Bi-directional, multi-layer recurrent layers
- Parameters of the convolutional layers: kernel_size, stride, dilation. It is often beneficial to do some dilation in later layers.
- Dropout, not only before the last layer but also between hidden layers
- On smaller tasks such as ours, deep convolutional models (if properly tuned) can give better

results than recurrent models

Try to be creative.

Copy/paste the class definition of the model (only). Please use a preformatted code block when pasting the code. Also, give the average dev accuracy of your model that you got using the `evalute_model_full(Model)` function (I will use it as a reference only, as I will rerun the evaluation myself).

```
class BestSadModel(nn.Module):

  def __init__(self, feature_dim):
    ...
```

Also, write a few sentences about what you tried, what kind of things didn't give any improvement and what was the most important part in your model to get high accuracy.

Grading: all models will be evaluated using the function implemented in the provided notebook (`evalute_model_full`) that initializes a model from scratch 3 times, and calculates the avegare best accuracy on dev data. Next, the ranking of the submitted models will be calculated and the points 1 to 10 will be distributed uniformly based on the rankings. E.g., if I will get 20 submissions, then the most accurate two models will get 10 points, next two will get 9 points, etc. In case of rounding, mathematical rounding to the nearest integer will be used.