# Creating a GRU model using Trax: Ungraded Lecture Notebook

For this lecture notebook you will be using Trax's layers. These are the building blocks for creating neural networks with Trax.

In [1]:
```
import trax
from trax import layers as tl
```

```
INFO:tensorflow:tokens_length=568 inputs_length=512 targets_length=114 noise_density=0.15 mean_noise_span_length=3.0
```

Trax allows to define neural network architectures by stacking layers (similarly to other libraries such as Keras). For this the `Serial()` is often used as it is a combinator that allows to stack layers serially using function composition.

Next you can see a simple vanilla NN architecture containing 1 hidden(dense) layer with 128 cells and output (dense) layer with 10 cells on which we apply the final layer of logsoftmax.

In [2]:
```
mlp = tl.Serial(
  tl.Dense(128),
  tl.Relu(),
  tl.Dense(10),
  tl.LogSoftmax()
)
```

Each of the layers within the `Serial` combinator layer is considered a sublayer. Notice that unlike similar libraries, **in Trax the activation functions are considered layers.** To know more about the `Serial` layer check the docs here.

You can try printing this object:

In [3]:
```
print(mlp)
```

```
Serial[
  Dense_128
  Relu
  Dense_10
  LogSoftmax
]
```

Printing the model gives you the exact same information as the model's definition itself.

By just looking at the definition you can clearly see what is going on inside the neural network. Trax is very straightforward in the way a network is defined, that is one of the things that makes it awesome!

## GRU MODEL

To create a `GRU` model you will need to be familiar with the following layers (Documentation link attached with each layer name):

- `ShiftRight` Shifts the tensor to the right by padding on axis 1. The `mode` should be specified and it refers to the context in which the model is being used. Possible values are: 'train', 'eval' or 'predict', predict mode is for fast inference. Defaults to "train".
- `Embedding` Maps discrete tokens to vectors. It will have shape `(vocabulary length X dimension of output vectors)`. The dimension of output vectors (also called `d_feature`) is the number of elements in the word embedding.
- `GRU` The GRU layer. It leverages another Trax layer called `GRUCell`. The number of GRU units should be specified and should match the number of elements in the word embedding. If you want to stack two consecutive GRU layers, it can be done by using python's list comprehension.
- `Dense` Vanilla Dense layer.
- `LogSoftMax` Log Softmax function.

Putting everything together the GRU model will look like this:

In [4]:
```
mode = 'train'
vocab_size = 256
model_dimension = 512
n_layers = 2

GRU = tl.Serial(
      tl.ShiftRight(mode=mode), # Do remember to pass the mode parameter if you are using it for interence/test as default is tr
      tl.Embedding(vocab_size=vocab_size, d_feature=model_dimension),
      [tl.GRU(n_units=model_dimension) for _ in range(n_layers)], # You can play around n_layers if you want to stack more GRU l
      tl.Dense(n_units=vocab_size),
      tl.LogSoftmax()
    )
```

Next is a helper function that prints information for every layer (sublayer within `Serial`):

*Try changing the parameters defined before the GRU model and see how it changes!*

In [5]:
```
def show_layers(model, layer_prefix="Serial.sublayers"):
    print(f"Total layers: {len(model.sublayers)}\n")
    for i in range(len(model.sublayers)):
        print('========')
        print(f'{layer_prefix}_{i}: {model.sublayers[i]}\n')

show_layers(GRU)
```

```
Total layers: 6
```

```
========
Serial.sublayers_0: ShiftRight(1)

========
Serial.sublayers_1: Embedding_256_512

========
Serial.sublayers_2: GRU_512

========
Serial.sublayers_3: GRU_512

========
Serial.sublayers_4: Dense_256

========
Serial.sublayers_5: LogSoftmax
```

Hope you are now more familiarized with creating GRU models using Trax.

You will train this model in this week's assignment and see it in action.

**GRU and the trax minions will return, in this week's endgame.**

In [ ]: