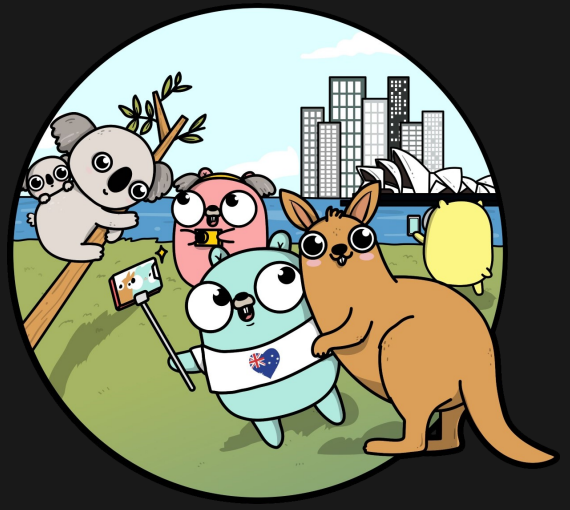
GETTING STARTED WITH DEEP LEARNING IN GO



Speaker

Darrell Chua

Senior Data Scientist
Kablamo
(Twitter: @cfgt)

ABOUT ME

- Data scientist for more than 10 years
- Have mostly been in the financial industry
- Have been programming in Go for 3 years

AGENDA

- What is Deep Learning?
- Building a model in Gorgonia
- How to port code from other popular libraries?

WHAT IS DEEP LEARNING?

- Machine Learning
- Artificial Neural Network
- Depth In Multiple Layers

WHEN TO USE DEEP LEARNING?

- You have large amounts of examples
- The model can be a black box
- You have the budget for data collection, model training and model execution
- There are features yet to be extracted from unstructured data

HOW LARGE?

- Ideally, tens or hundreds of thousands of labeled examples
- For deep learning to really shine, you want millions of records
- But really, it depends

HOW LARGE?

“As one Google Translate engineer put it, "when you go from 10,000 training examples to 10 billion training examples, it all starts to work. Data trumps everything.”

— Garry Kasparov, Deep Thinking: Where Machine Intelligence Ends and Human Creativity Begins

WHAT DO WE NEED?

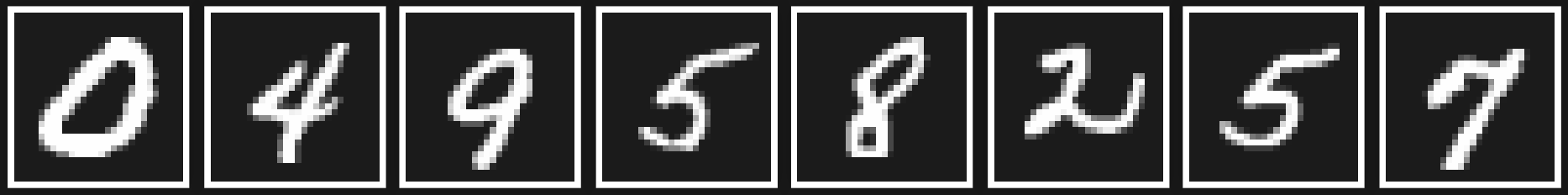
- We need to define the problem.
- We need to find suitable data.
- We need to define a measure of success.

THE PROBLEM

Classic Example:
Recognising handwritten digits

THE DATA: MNIST

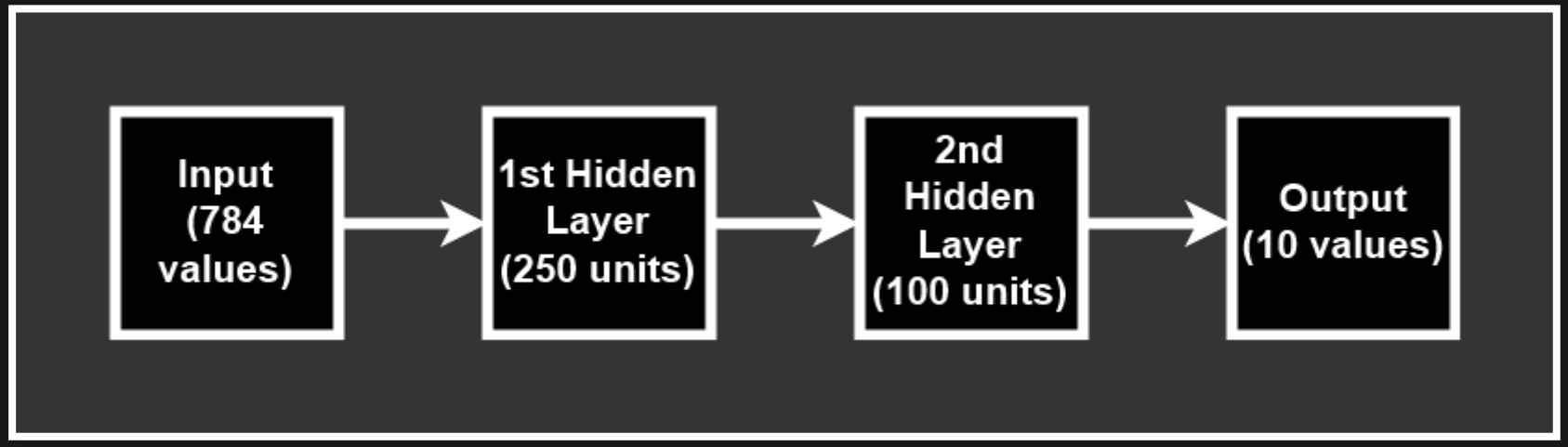
Images of handwritten digits that are 28 by 28



SUCCESS

For this example, let's just try to hit >90% accuracy.

WHAT DOES OUR MODEL LOOK LIKE?



WHAT IS A HIDDEN LAYER?

- A layer is typically a high-level way of referring to a set of nodes
- Typically, it will contain a set of weights and a bias term

$$y = w \cdot x + b$$

WHAT IS A HIDDEN LAYER?

- It will then usually end with an activation function node - usually not entirely linear in order to ensure that the resulting model is non-linear
- For this example, we will be using the ReLU activation function:

$$R(x) = max(0, x)$$

WHAT IS A HIDDEN LAYER?

- It will then usually end with an activation function node - usually not entirely linear in order to ensure that the resulting model is non-linear
- For this example, we will be using the ReLU activation function:

$$R(x) = \left\{ \begin{array}{ll} x & x > 0 \\ 0 & x <= 0 \end{array} \right\}$$

OUTPUT LAYER

- We are going to convert our labels to be one-hot encoded.
- We'll be using softmax for output:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

IMPLEMENTATION - WEIGHTS

```
w1 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(784,250),
    gorgonia.WithName("w1"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
b1 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(1,250),
    gorgonia.WithName("b1"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
```

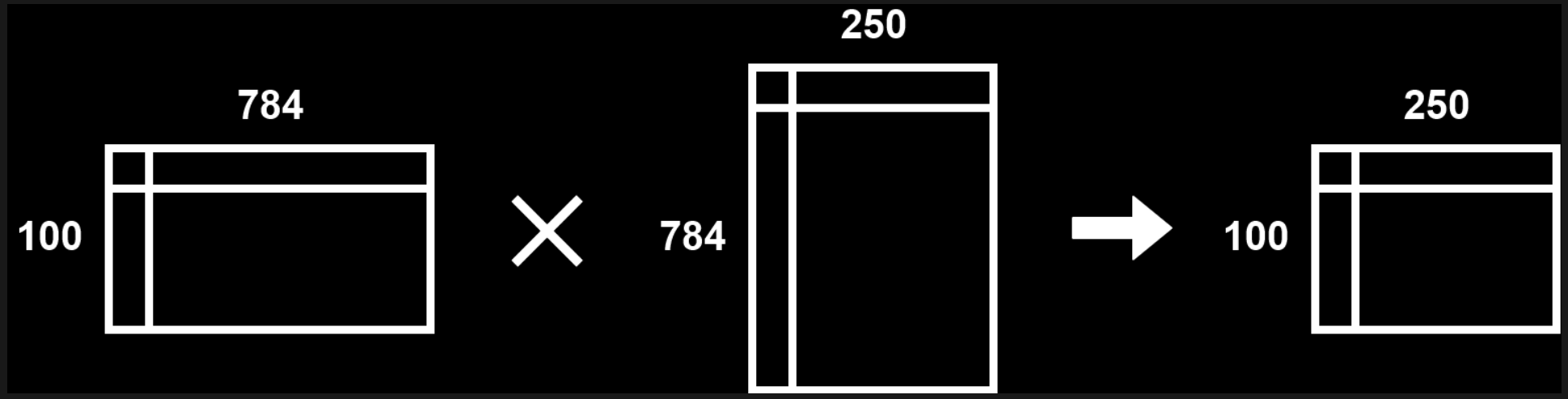
IMPLEMENTATION - WEIGHTS

```
w2 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(250,100),
    gorgonia.WithName("w2"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
b2 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(1,100),
    gorgonia.WithName("b2"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
```

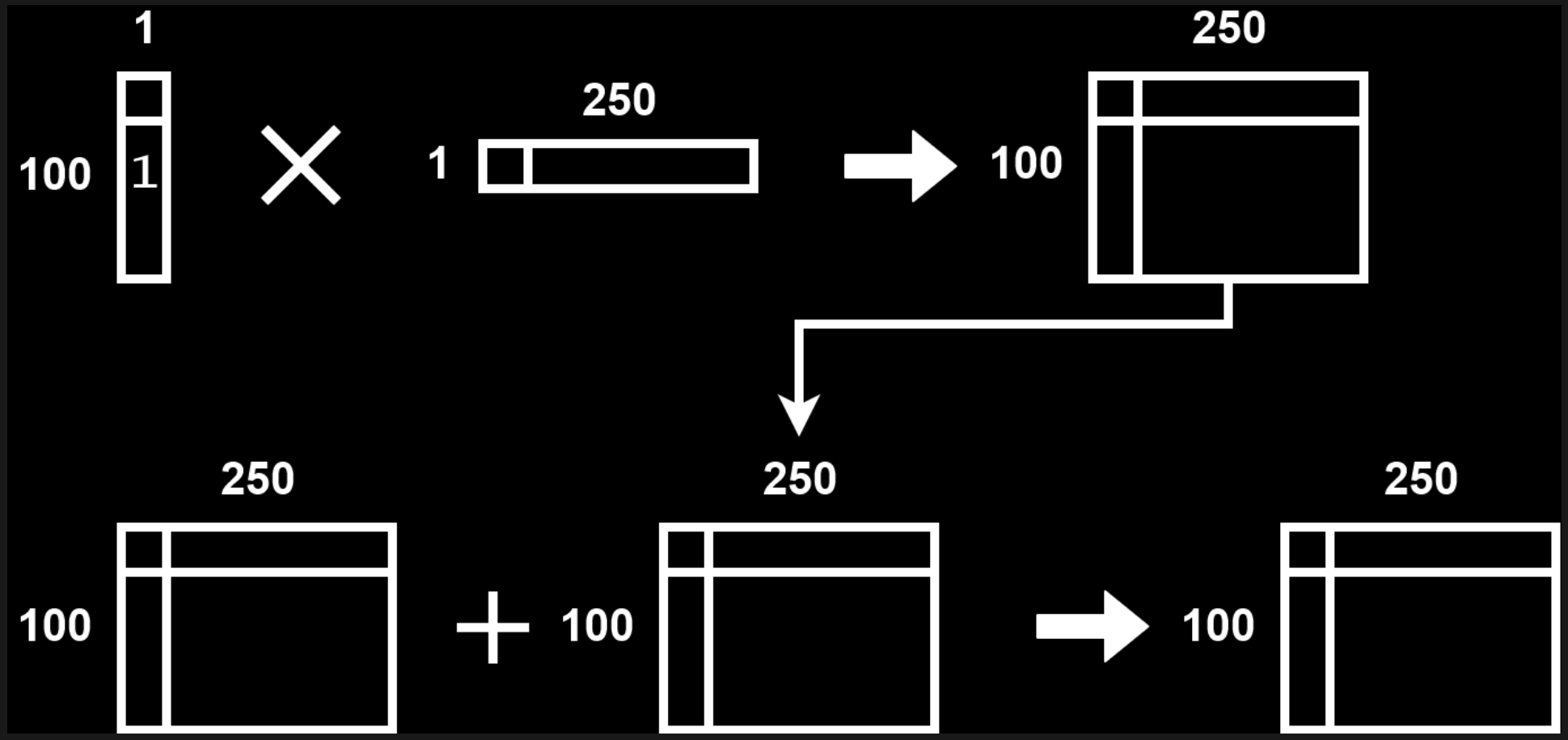
IMPLEMENTATION - WEIGHTS

```
w3 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(100,10),
    gorgonia.WithName("w3"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
b3 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(1,10),
    gorgonia.WithName("b3"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
```

IMPLEMENTATION - LAYERS



IMPLEMENTATION - LAYERS



IMPLEMENTATION - LAYERS

```
l0 = input
```

IMPLEMENTATION - LAYERS

```
l1 = gorgonia.Must(gorgonia.Rectify(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l0,m.w1)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b1))
    ))
))
```

IMPLEMENTATION - LAYERS

```
l2 = gorgonia.Must(gorgonia.Rectify(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l1,m.w2)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b2))
    ))
))
```

IMPLEMENTATION - LAYERS

```
l3 = gorgonia.Must(gorgonia.Softmax(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l2,m.w3)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b3))
    ))
))
m.out = l3
```

IMPLEMENTATION - CROSS ENTROPY LOSS

$$-\frac{1}{N}\sum_{N}\sum_{j} y_j \log(p_j)$$

N is the number of items in the batch

log is the natural log

y_j is a binary indicator for the correct classification

p_j is the predicted probability the observation o is of class c

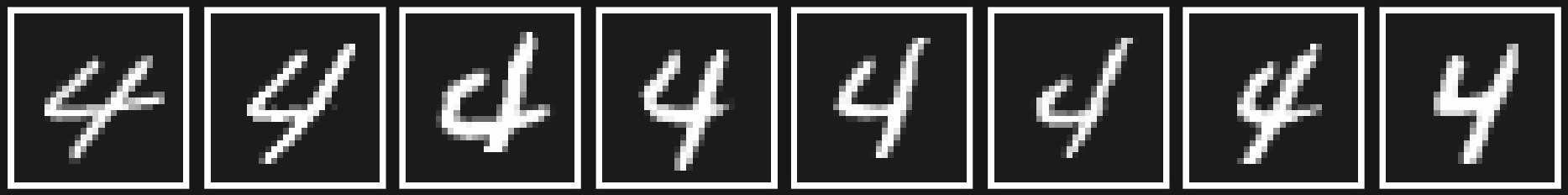
IMPLEMENTATION - LOSS

```
losses:= gorgonia.Must(gorgonia.Neg(
    gorgonia.Must(gorgonia.Mean(
        gorgonia.Must(gorgonia.HadamardProd(
            gorgonia.Must(gorgonia.Log(m.out)),
            y,
        )),
    )),
))
```

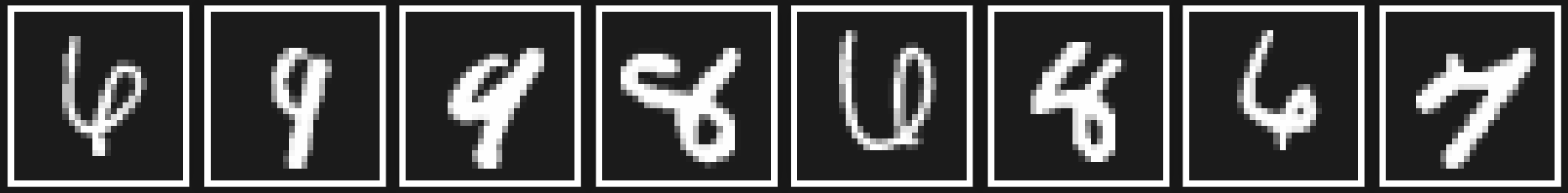
IMPLEMENTATION - ADMIN

```
vm := gorgonia.NewTapeMachine(
    g,
    gorgonia.BindDualValues(m.learnables()...),
)
solver := gorgonia.NewRMSPropSolver(
    gorgonia.WithBatchSize(float64(bs)),
)
```

RESULTS - CORRECT



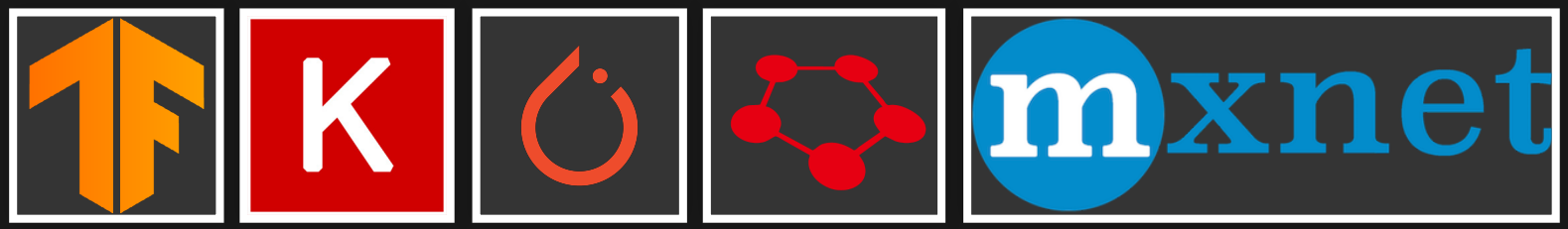
RESULTS - WRONG



RESULTS - ACCURACY

When this was allowed to run, it was correct 98.16% of the time on the test set.

POPULAR LIBRARIES IN OTHER LANGUAGES



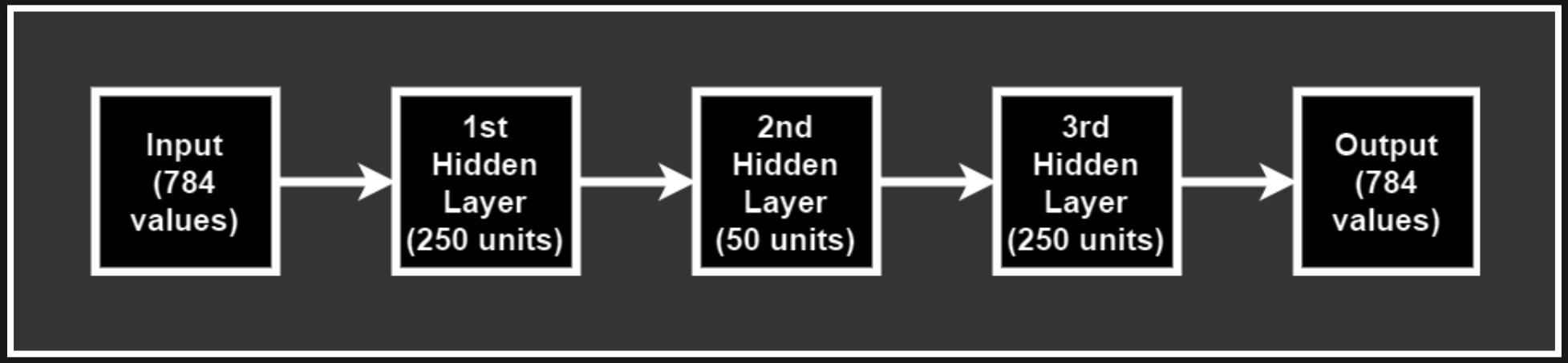
WE WRITE GO, SO THERE'S ONLY ONE REAL OPTION

(Golgi coming soon™)

AUTOENCODERS

- Let's look at an example of porting some code to Gorgonia
- There are many examples on the Internet for building autoencoders in TensorFlow and PyTorch
- Let's pick something that looks sort of similar and try to work it out

AUTOENCODERS



STRUCTURE - TENSORFLOW

```
weights = {
    'encoder_h1': tf.Variable(tf.random_normal([784,250])),
    'encoder_h2': tf.Variable(tf.random_normal([250,50])),
    'decoder_h1': tf.Variable(tf.random_normal([50,250])),
    'decoder_h2': tf.Variable(tf.random_normal([250,784])),
}
biases = {
    'encoder_b1': tf.Variable(tf.random_normal([250])),
    'encoder_b2': tf.Variable(tf.random_normal([50])),
    'decoder_b1': tf.Variable(tf.random_normal([250])),
    'decoder_b2': tf.Variable(tf.random_normal([784])),
}
```

STRUCTURE - KERAS

```
input_img = Input(shape=(784,))
encoded = Dense(250, activation='relu')(input_img)
encoded = Dense(50, activation='relu')(encoded)

decoded = Dense(250, activation='relu')(encoded)
decoded = Dense(784, activation='sigmoid')(decoded)
```

STRUCTURE - PYTORCH

```
def __init__(self):
    super(AutoEncoder, self).__init__()

    self.e1 = nn.Linear(784,250)
    self.e2 = nn.Linear(250,50)

    self.d1 = nn.Linear(50,250)
    self.d2 = nn.Linear(250,784)
```

STRUCTURE EXAMPLE - GORGONIA

```
w1 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(784,250),
    gorgonia.WithName("w1"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
b1 := gorgonia.NewMatrix(
    g,
    dt,
    gorgonia.WithShape(1,250),
    gorgonia.WithName("b1"),
    gorgonia.WithInit(gorgonia.GlorotN(1.0))
)
```

ENCODER - TENSORFLOW

```
layer_1 = tf.nn.relu(
    tf.add(
        tf.matmul(x, weights['encoder_h1']),
        biases['encoder_b1']
    )
)
layer_2 = tf.nn.relu(
    tf.add(
        tf.matmul(layer_1,weights['encoder_h2']),
        biases['encoder_b2']
    )
)
```

DECODER - TENSORFLOW

```
layer_3 = tf.nn.relu(
    tf.add(
        tf.matmul(layer_2, weights['decoder_h1']),
        biases['decoder_b1']
    )
)
layer_4 = tf.nn.sigmoid(
    tf.add(
        tf.matmul(layer_3, weights['decoder_h2']),
        biases['decoder_b2']
    )
)
```

ENCODER + DECODER - KERAS

```
input_img = Input(shape=(784,))
encoded = Dense(250, activation='relu')(input_img)
encoded = Dense(50, activation='relu')(encoded)

decoded = Dense(250, activation='relu')(encoded)
decoded = Dense(784, activation='sigmoid')(decoded)

autoencoder = Model(input_img, decoded)
```

ENCODER + DECODER - PYTORCH

```
def forward(self,x):
    x = F.relu(self.e1(x))
    x = F.relu(self.e2(x))

    x = F.relu(self.d1(x))
    x = F.sigmoid(self.d2(x))

    return x
```

ENCODER - GORGONIA

```
l1 = gorgonia.Must(gorgonia.Rectify(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l0,m.w1)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b1))
    ))
))
l2 = gorgonia.Must(gorgonia.Rectify(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l1,m.w2)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b2))
    ))
))
```

DECODER - GORGONIA

```
l3 = gorgonia.Must(gorgonia.Rectify(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l2,m.w3)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b3))
    ))
))
l4 = gorgonia.Must(gorgonia.Sigmoid(
    gorgonia.Must(gorgonia.Add(
        gorgonia.Must(gorgonia.Mul(l3,m.w4)),
        gorgonia.Must(gorgonia.Mul(m.mOnes,m.b4))
    ))
))
```

ENCODER + DECODER - GOLGI

```
autoencoder := G.ComposeSeq(
    x,
    G.L(G.ConsFC(G.WithShape(250), G.WithName("e1"),
        G.WithActivation(GG.Rectify))),
    G.L(G.ConsFC(G.WithShape(50), G.WithName("e2"),
        G.WithActivation(GG.Rectify))),
    G.L(G.ConsFC(G.WithShape(250), G.WithName("d1"),
        G.WithActivation(GG.Rectify))),
    G.L(G.ConsFC(G.WithShape(784), G.WithName("d2"),
        G.WithActivation(GG.Sigmoid))),
)
```

LOSS - TENSORFLOW

```
loss = tf.reduce_mean(tf.pow(y_true - y_pred, 2))
```

LOSS - KERAS

```
autoencoder.compile(loss='mean_squared_error')
```

LOSS - PYTORCH

```
loss_func = nn.MSELoss()
```

LOSS - MATH

$$\mathrm{MSE} = \frac{1}{N}\sum_{N}(y - \hat{y})^2$$

LOSS - GORGONIA

```
losses = gorgonia.Must(gorgonia.Mean(
    gorgonia.Must(gorgonia.Square(
        gorgonia.Must(gorgonia.Sub(y, m.out))
    ))
))
```

AUTOENCODER RESULTS

SHAMELESS PLUG

- Twitter:**@cfgt**
- Buy my book: http://tiny.cc/HODLinGo