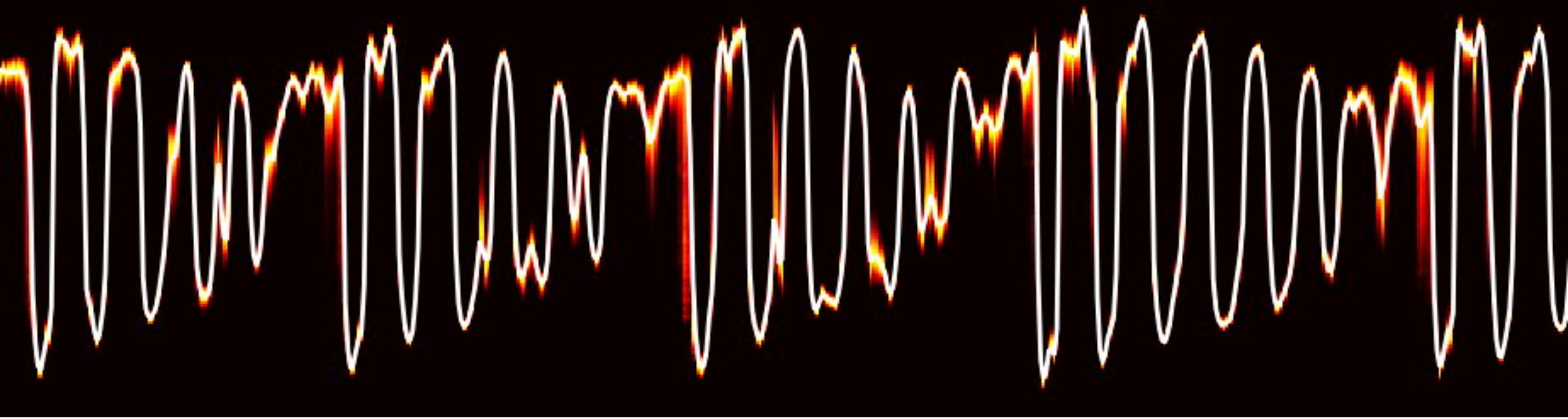
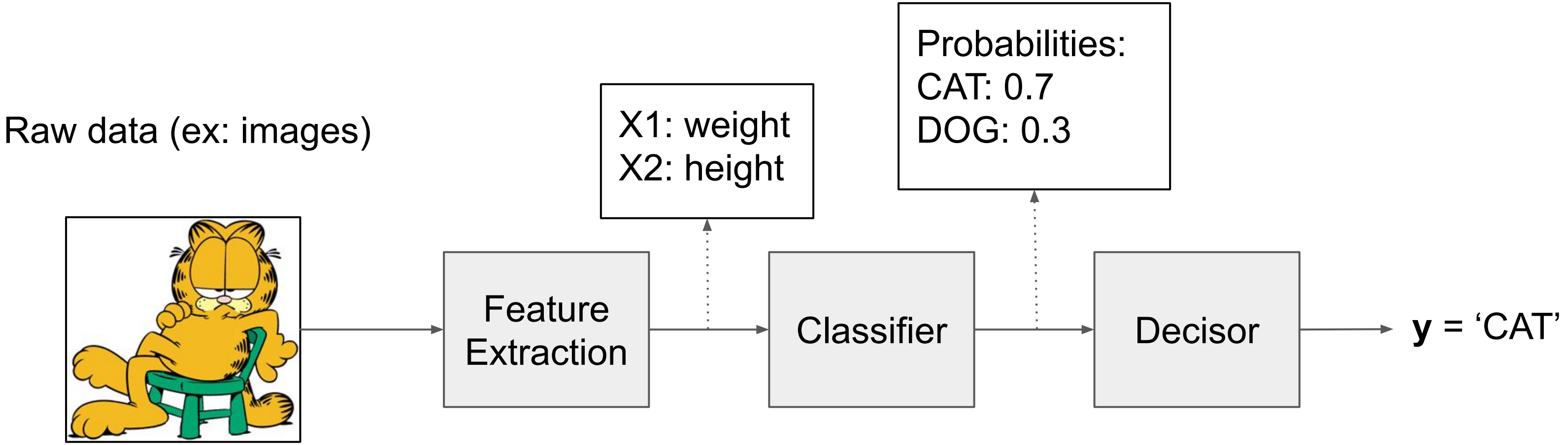


Intro to Speaker Recognition with PyTorch

Santiago Pascual de la Puente
santi.pascual@upc.edu

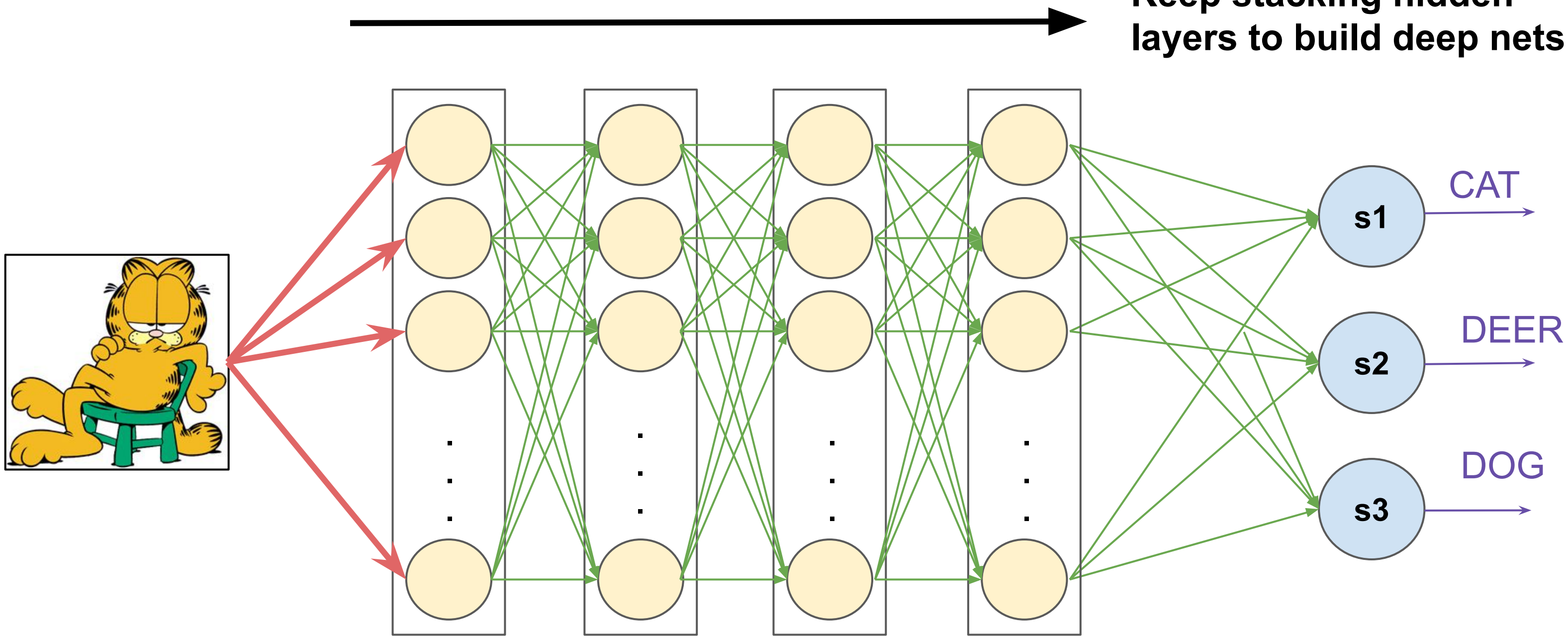
PhD Candidate
Universitat Politècnica de Catalunya
Technical University of Catalonia

Classic Machine Learning classification pipeline

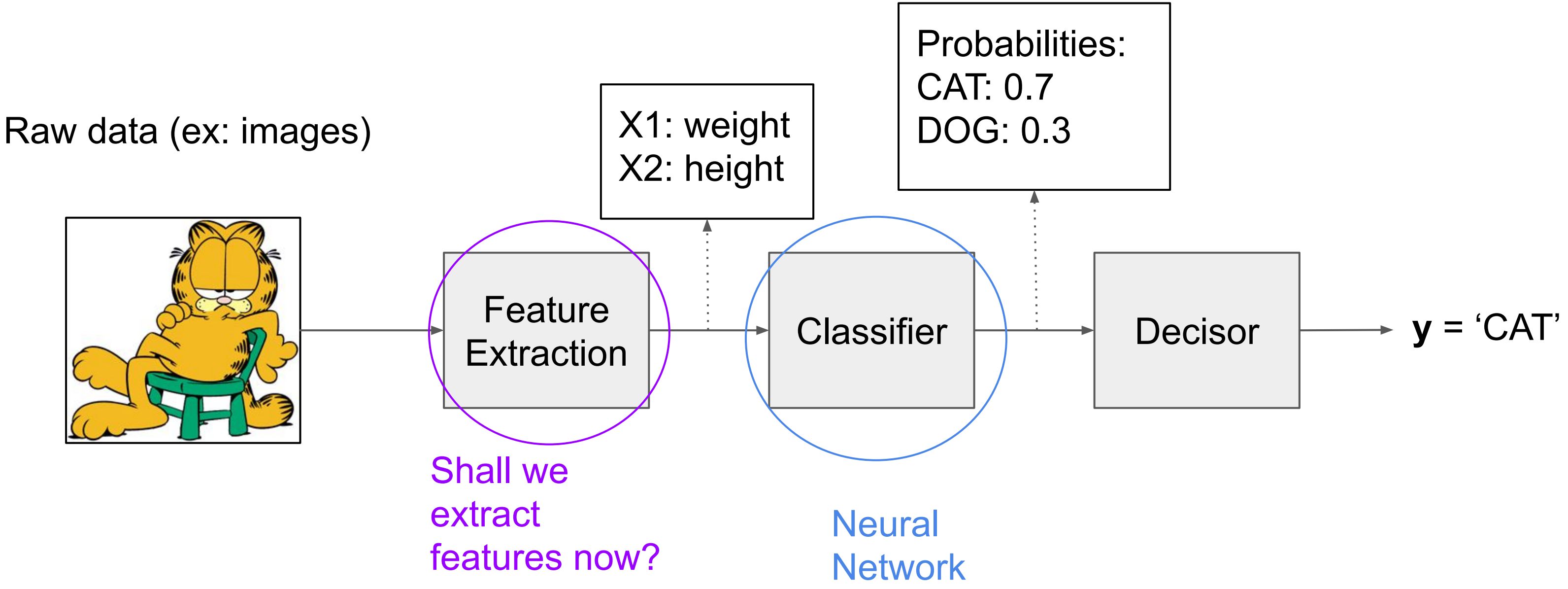


Going deeper: what neural networks is about

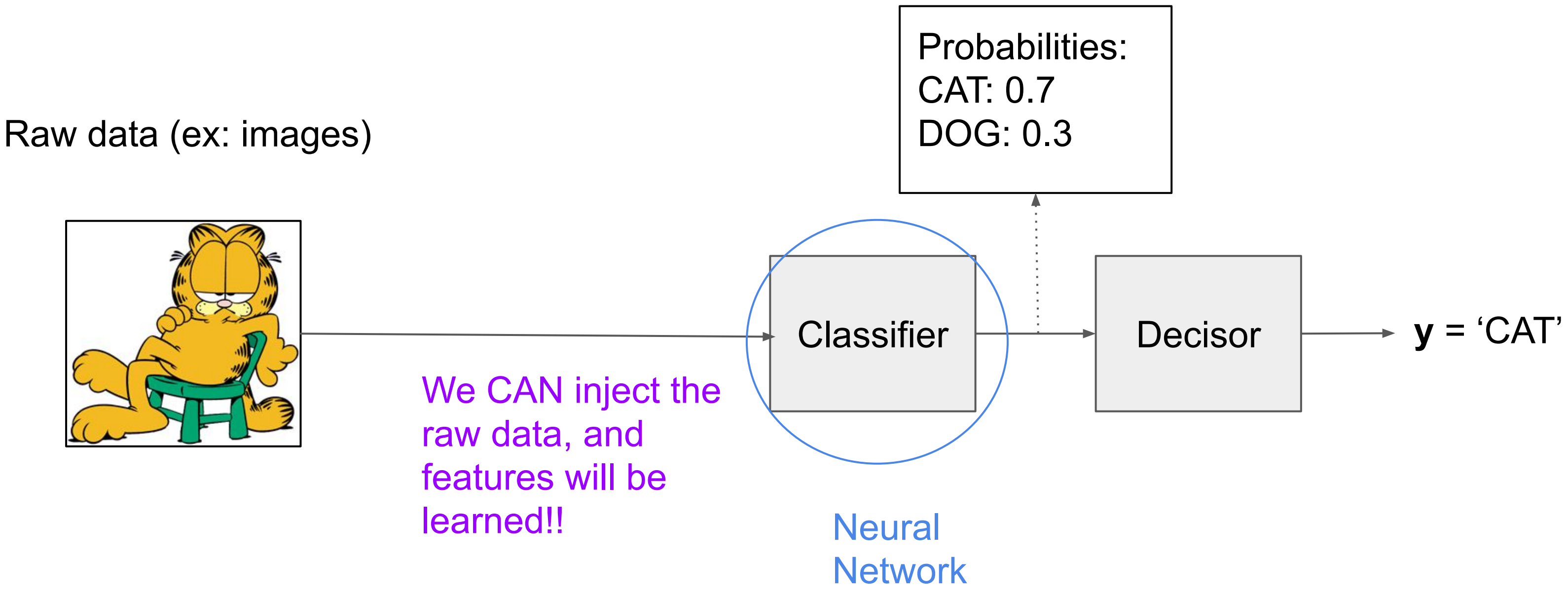
Keep stacking hidden layers to build deep nets



Classic Machine Learning classification pipeline



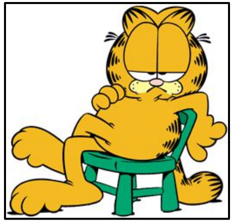
Classic Machine Learning classification pipeline



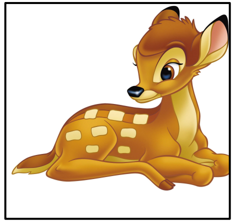
End to End concept

Multi-class classification labels with neurons?

Example: we have RGB coloured images of **cats**, **dogs**, and **deers**. Each image is 32x32 pixels, with a label per image. Use **neurons** to classify with **one-hot codes**.



[1, 0, 0]



[0, 1, 0]

[0, 0, 1]

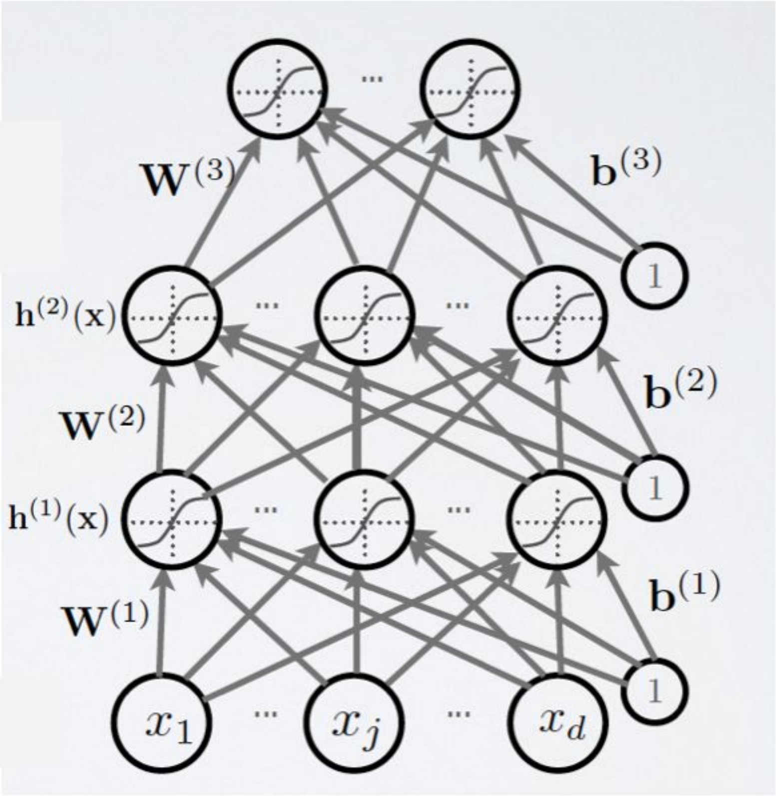
The Neural Network

The i-th layer is then defined by a matrix **Wi** and a vector **bi,** and the activation is simply a dot product plus **bi**:

$$h_i = f(W_i \cdot h_{i-1} + b_i)$$

Num parameters to learn at i-th layer:

$$N^i_{params} = N^i_{inputs} \times N^i_{units} + N^i_{units}$$



Slide Credit: Hugo Laroche NN course

The Deep Learning Framework

PyTorch

- **Provides GPU computation**
- **Does all the back-propagation work for you! (you write no derivative code)**

https://pytorch.org/get-started/locally/

PyTorch

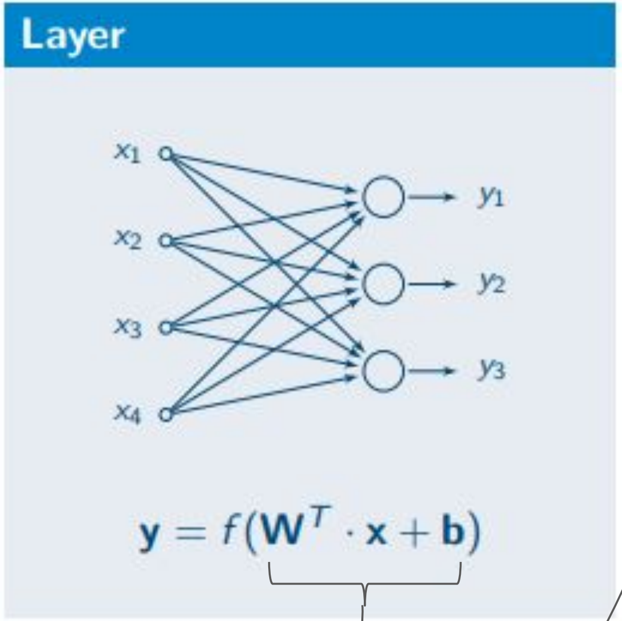
START LOCALLY

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, 1.3 builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. Anaconda is our recommended package manager since it installs all dependencies. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

PyTorch Build	Stable (1.3)	Preview (Nightly)			
Your OS	Linux	Mac	Windows		
Package	Conda	Pip	LibTorch	Source	
Language	Python 2.7	Python 3.5	Python 3.6	Python 3.7	C++
CUDA	9.2	10.1	None		
Run this Command:	`conda install pytorch torchvision cudatoolkit=10.1 -c pytorch`				

https://pytorch.org/get-started/locally/

Fully Connected Layer



CLASS `torch.nn.Linear(in_features, out_features, bias=True)` [SOURCE]

Applies a linear transformation to the incoming data: $y = xA^T + b$

Parameters:

- **in_features** – size of each input sample
- **out_features** – size of each output sample
- **bias** – If set to False, the layer will not learn an additive bias. Default: `True`

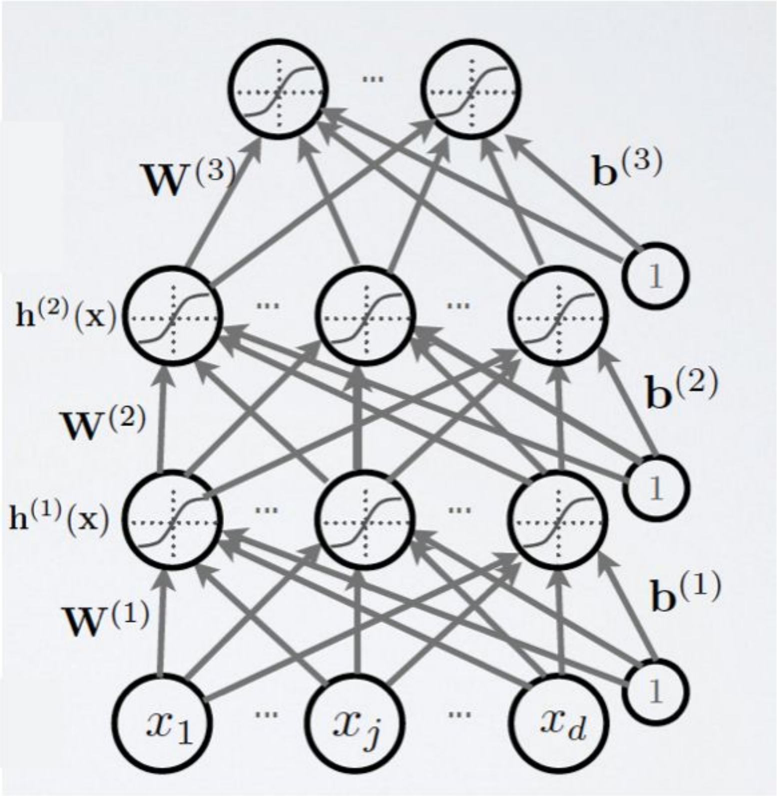
Shape:

- Input: $(N, *, \text{in\_features})$ where $*$ means any number of additional dimensions
- Output: $(N, *, \text{out\_features})$ where all but the last dimension are the same shape as the input.

Variables:

- **weight** – the learnable weights of the module of shape $(\text{out\_features}, \text{in\_features})$. The values are initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = \frac{1}{\text{in\_features}}$
- **bias** – the learnable bias of the module of shape (out_features). If `bias` is `True`, the values are initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ where $k = \frac{1}{\text{in\_features}}$

Fully Connected: MultiLayer Perceptron



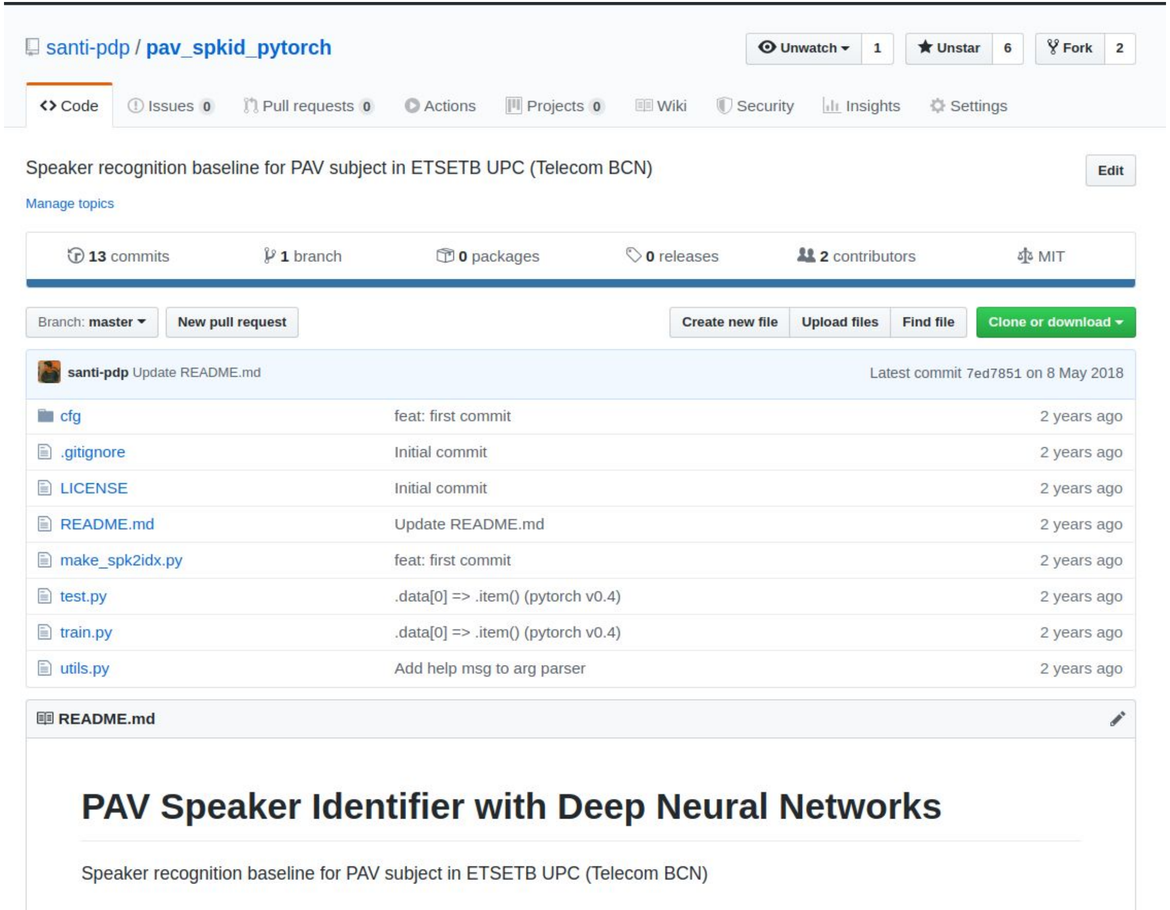
Slide Credit: Hugo Laroche NN course

Many fully connected layers with many units.

```
NUM_INPUTS=100
HIDDEN_SIZE=1024
NUM_OUTPUTS=20

mlp = nn.Sequential(
      nn.Linear(NUM_INPUTS, HIDDEN_SIZE),
      nn.Tanh(),
      nn.Linear(HIDDEN_SIZE, HIDDEN_SIZE),
      nn.Tanh(),
      nn.Linear(HIDDEN_SIZE, NUM_OUTPUTS),
      nn.LogSoftmax(dim=1)
)
```

PAV SpkID Public Repo here



Training Arguments Review

```
parser.add_argument('--db_path', type=str, default='mcp',
                    help='path to feature files (default: ./mcp)')
parser.add_argument('--tr_list_file', type=str, default='cfg/all.train',
                    help='File list of train files (default: cfg/all.train)')
parser.add_argument('--va_list_file', type=str, default='cfg/all.test',
                    help='File list of eval files (default: cfg/all.test)')
parser.add_argument('--ext', type=str, default='mcp',
                    help='Extension of feature files (default mcp)')
parser.add_argument('--spk2idx', type=str, default='cfg/spk2idx.json',
                    help='File to map spk code to spkID: 0,1, .... (def. cfg/spk2idx.json)')
parser.add_argument('--batch_size', type=int, default=1000, help='batch size (default: 1000)')
parser.add_argument('--hsize', type=int, default=100,
                    help='Num. of units in hidden layers (default=100)')
parser.add_argument('--in_frames', type=int, default=21,
                    help='num of frames stacked to create the input features (default: 21)')
parser.add_argument('--patience', type=int, default=10,
                    help='Num of epochs to wait if val loss improves '
                         '(default: 10)')
parser.add_argument('--lr', type=float, default=0.001, help='Learning rate (def. 0.001)')
parser.add_argument('--momentum', type=float, default=0.5, help='Momentum (def. 0.5)')
parser.add_argument('--epoch', type=int, default=20,
                    help='Number of epochs to train (default: 20)')
parser.add_argument('--log_freq', type=int, default=20,
                    help='Every <log_freq> batches, log stuff (default: 20)')
parser.add_argument('--save_path', type=str, default='ckpt', help='path for the model (def. ckpt)')
```

The NNet defined in PAV SpkID

```
# Feed Forward Neural Network
model = nn.Sequential(nn.Linear(dset.input_dim * dset.in_frames, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, dset.num_spks),
                      nn.LogSoftmax(dim=1))
```

The NNet defined in PAV SpkID

You can play with the hidden size of the network in the --hsize command line argument

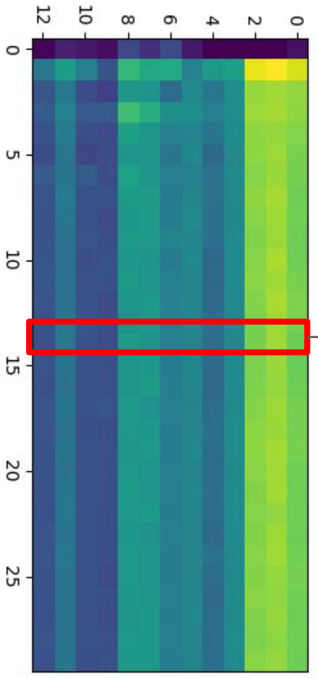
```
# Feed Forward Neural Network
model = nn.Sequential(nn.Linear(dset.input_dim * dset.in_frames, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, opts.hsize),
                      nn.ReLU(),
                      nn.Linear(opts.hsize, dset.num_spks),
                      nn.LogSoftmax(dim=1))
```

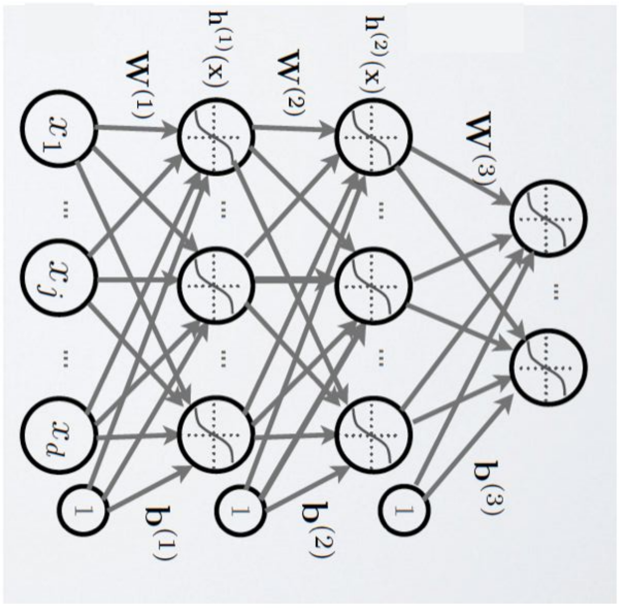
Training Arguments Review

```python
parser.add_argument('--db_path', type=str, default='mcp',
                    help='path to feature files (default: ./mcp)')
parser.add_argument('--tr_list_file', type=str, default='cfg/all.train',
                    help='File list of train files (default: cfg/all.train)')
parser.add_argument('--va_list_file', type=str, default='cfg/all.test',
                    help='File list of eval files (default: cfg/all.test)')
parser.add_argument('--ext', type=str, default='mcp',
                    help='Extension of feature files (default mcp)')
parser.add_argument('--spk2idx', type=str, default='cfg/spk2idx.json',
                    help='File to map spk code to spkID: 0,1, .... (def. cfg/spk2idx.json)')
parser.add_argument('--batch_size', type=int, default=1000, help='batch size (default: 1000)')
parser.add_argument('--hsize', type=int, default=100,
                    help='Num. of units in hidden layers (default=100)')
parser.add_argument('--in_frames', type=int, default=21,
                    help='num of frames stacked to create the input features (default: 21)')
parser.add_argument('--patience', type=int, default=10,
                    help='Num of epochs to wait if val loss improves '
                         '(default: 10)')
parser.add_argument('--lr', type=float, default=0.001, help='Learning rate (def. 0.001)')
parser.add_argument('--momentum', type=float, default=0.5, help='Momentum (def. 0.5)')
parser.add_argument('--epoch', type=int, default=20,
                    help='Number of epochs to train (default: 20)')
parser.add_argument('--log_freq', type=int, default=20,
                    help='Every <log_freq> batches, log stuff (default: 20)')
parser.add_argument('--save_path', type=str, default='ckpt', help='path for the model (def. ckpt)')
```

About input context

One input frame in the context with 13 MFCC: 13x1 features



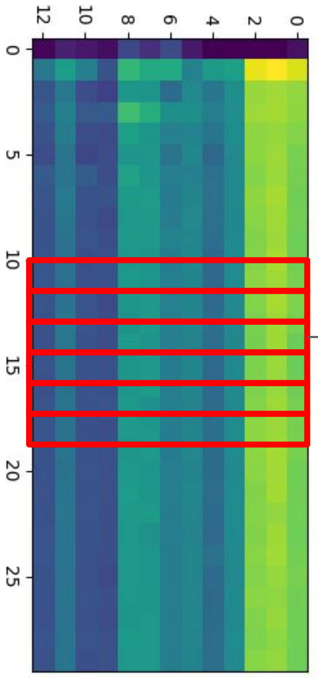


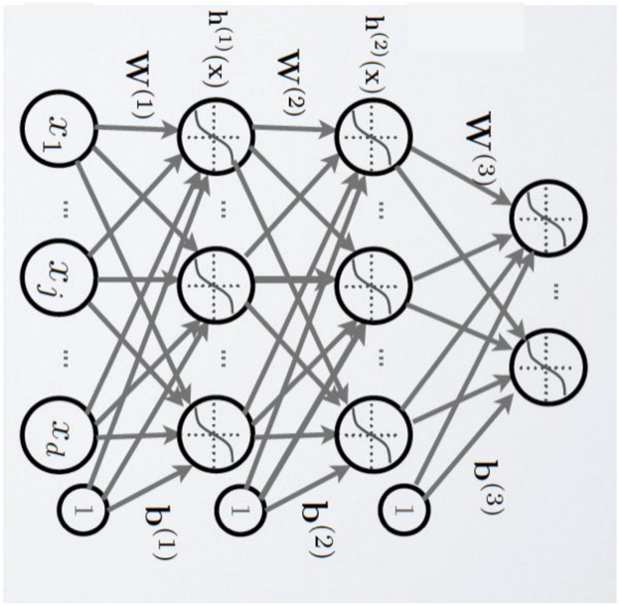
[0.1 0.1 0.08 0.02 ... $\boxed{0.7}$]

Max index

About input context

21 input frames in the context with 13 MFCC: 13x21 features





[0.07 0.09 0.05 0.01 ... 0.8]

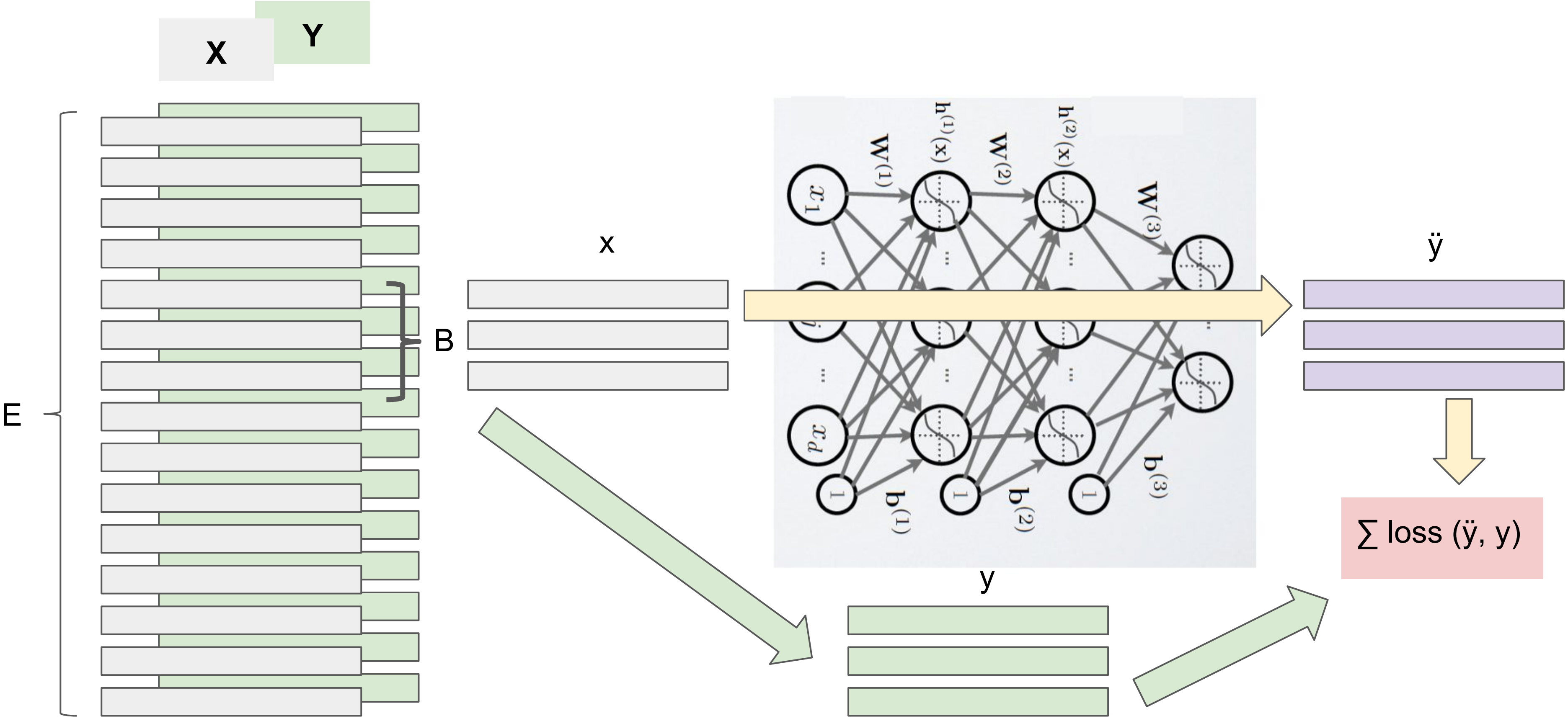
Max index

Training Arguments Review

```
parser.add_argument('--db_path', type=str, default='mcp',
                    help='path to feature files (default: ./mcp)')
parser.add_argument('--tr_list_file', type=str, default='cfg/all.train',
                    help='File list of train files (default: cfg/all.train)')
parser.add_argument('--va_list_file', type=str, default='cfg/all.test',
                    help='File list of eval files (default: cfg/all.test)')
parser.add_argument('--ext', type=str, default='mcp',
                    help='Extension of feature files (default mcp)')
parser.add_argument('--spk2idx', type=str, default='cfg/spk2idx.json',
                    help='File to map spk code to spkID: 0,1, .... (def. cfg/spk2idx.json)')
parser.add_argument('--batch_size', type=int, default=1000, help='batch size (default: 1000)')
parser.add_argument('--hsize', type=int, default=100,
                    help='Num. of units in hidden layers (default=100)')
parser.add_argument('--in_frames', type=int, default=21,
                    help='num of frames stacked to create the input features (default: 21)')
parser.add_argument('--patience', type=int, default=10,
                    help='Num of epochs to wait if val loss improves '
                         '(default: 10)')
parser.add_argument('--lr', type=float, default=0.001, help='Learning rate (def. 0.001)')
parser.add_argument('--momentum', type=float, default=0.5, help='Momentum (def. 0.5)')
parser.add_argument('--epoch', type=int, default=20,
                    help='Number of epochs to train (default: 20)')
parser.add_argument('--log_freq', type=int, default=20,
                    help='Every <log_freq> batches, log stuff (default: 20)')
parser.add_argument('--save_path', type=str, default='ckpt', help='path for the model (def. ckpt)')
```

Mini(batches) and epochs

E: Samples for 1 epoch
B: Samples for 1 minibatch



Training Arguments Review

```python
parser.add_argument('--db_path', type=str, default='mcp',
                    help='path to feature files (default: ./mcp)')
parser.add_argument('--tr_list_file', type=str, default='cfg/all.train',
                    help='File list of train files (default: cfg/all.train)')
parser.add_argument('--va_list_file', type=str, default='cfg/all.test',
                    help='File list of eval files (default: cfg/all.test)')
parser.add_argument('--ext', type=str, default='mcp',
                    help='Extension of feature files (default mcp)')
parser.add_argument('--spk2idx', type=str, default='cfg/spk2idx.json',
                    help='File to map spk code to spkID: 0,1, .... (def. cfg/spk2idx.json)')
parser.add_argument('--batch_size', type=int, default=1000, help='batch size (default: 1000)')
parser.add_argument('--hsize', type=int, default=100,
                    help='Num. of units in hidden layers (default=100)')
parser.add_argument('--in_frames', type=int, default=21,
                    help='num of frames stacked to create the input features (default: 21)')
parser.add_argument('--patience', type=int, default=10,
                    help='Num of epochs to wait if val loss improves '
                         '(default: 10)')
parser.add_argument('--lr', type=float, default=0.001, help='Learning rate (def. 0.001)')
parser.add_argument('--momentum', type=float, default=0.5, help='Momentum (def. 0.5)')
parser.add_argument('--epoch', type=int, default=20,
                    help='Number of epochs to train (default: 20)')
parser.add_argument('--log_freq', type=int, default=20,
                    help='Every <log_freq> batches, log stuff (default: 20)')
parser.add_argument('--save_path', type=str, default='ckpt', help='path for the model (def. ckpt)')
```

Choosing the learning rate

For **first order** optimization methods, we need to choose a learning rate (aka **step size**)

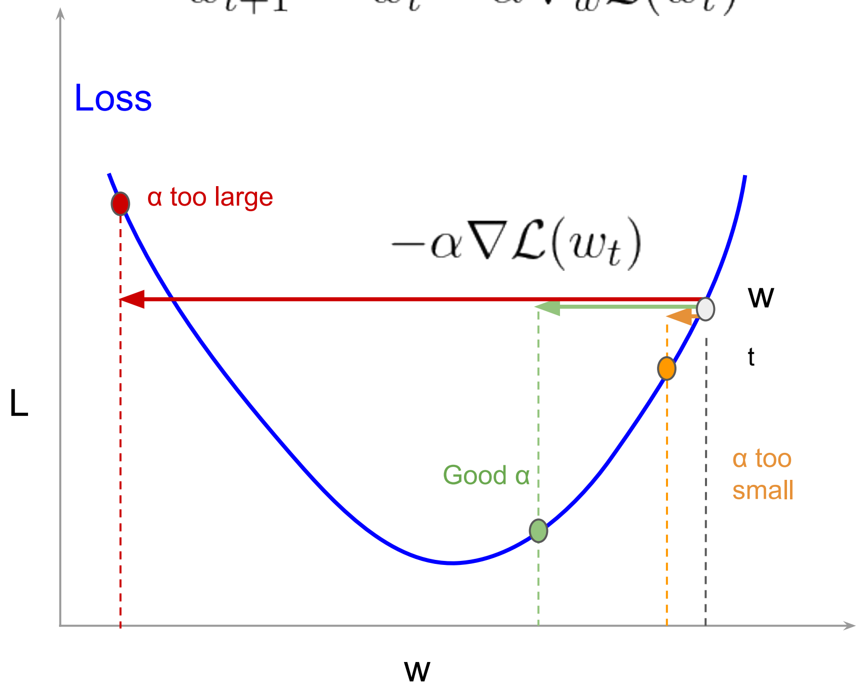
- **Too large**: overshoots local minimum, loss increases
- **Too small**: makes very slow progress, can get stuck
- **Good learning rate**: makes steady progress toward local minimum

Usually want a higher learning rate at the start and a lower one later on.

Common strategy in practice:

- Start off with a high LR (like 0.1 - 0.001),
- Run for several **epochs** (1 - 10)
- Decrease LR by multiplying a constant factor (0.1 - 0.5)

$$w_{t+1} = w_t - \alpha \nabla_w \mathcal{L}(w_t)$$



Slide credit: Elisa Sayrol