# Hyperparameters

**Learning Rate**

A picture containing text

Description automatically generated

**Mini Batch Size**

Too small could be too slow, too large could be computationally taxing and could result in worse accuracy. In practice small mini batch size have more noise in their error calculations and this noise is often helpful in preventing the training process from stopping at the local minima on the error curve, rather than the global minima that creates the best model.

32, 64,128, 256 are potentially good starting values to experiment with

**Number of epochs**

The number of training iterations is a hyperparameter we can optimize automatically using a technique called early stopping (also "early termination").

**Validation Monitor (Deprecated)**

In tensorflow, we can use a [ValidationMonitor with tf.contrib.learn](https://www.tensorflow.org/get_started/monitors#early_stopping_with_validationmonitor) to not only monitor the progress of training, but to also stop the training when certain conditions are met.

The following example from the ValidationMonitor documentation shows how to set it up. Note that the last three parameters indicate which metric we're optimizing.

validation\_monitor = tf.contrib.learn.monitors.ValidationMonitor(

test\_set.data,

test\_set.target,

every\_n\_steps=50,

metrics=validation\_metrics,

early\_stopping\_metric="loss",

early\_stopping\_metric\_minimize=True,

early\_stopping\_rounds=200)

The last parameter indicates to ValidationMonitor that it should stop the training process if the loss did not decrease in 200 steps (rounds) of training.

The validation\_monitor is then passed to tf.contrib.learn's "fit" method which runs the training process:

classifier = tf.contrib.learn.DNNClassifier(

feature\_columns=feature\_columns,

hidden\_units=[10, 20, 10],

n\_classes=3,

model\_dir="/tmp/iris\_model",

config=tf.contrib.learn.RunConfig(save\_checkpoints\_secs=1))

classifier.fit(x=training\_set.data,

y=training\_set.target,

steps=2000,

monitors=[validation\_monitor])

**SessionRunHook**

More recent versions of TensorFlow deprecated monitors in favor of [SessionRunHooks](https://www.tensorflow.org/api_docs/python/tf/train/SessionRunHook). SessionRunHooks are an evolving part of tf.train, and going forward appear to be the proper place where you'd implement early stopping.

At the time of writing, two pre-defined stopping monitors exist as a part of tf.train's [training hooks](https://www.tensorflow.org/api_guides/python/train#Training_Hooks):

* [StopAtStepHook](https://www.tensorflow.org/api_docs/python/tf/train/StopAtStepHook): A monitor to request the training stop after a certain number of steps
* [NanTensorHook](https://www.tensorflow.org/api_docs/python/tf/train/NanTensorHook): a monitor that monitor's loss and stops training if it encounters a NaN loss

NEXT

**Number of hidden units/layers**

The more hidden units the more is the learning capacity, but if we provide the model with too much capacity it might tend to overfit and just try to memorize the training set. If the model is overfitting(the training accuracy is much larger than the validation accuracy) reduce the hidden units. We can also use regularisation techniques such as dropout or L2 regularisation.

In practice it is often the case that 3-layer neural networks will outperform 2-layer nets, but going even deeper (4,5,6-layer) rarely helps much more. This is in stark contrast to Convolutional Networks, where depth has been found to be an extremely important component for a good recognition system (e.g. on order of 10 learnable layers).

**LSTM Vs GRU**

"These results clearly indicate the advantages of the gating units over the more traditional recurrent units. Convergence is often faster, and the final solutions tend to be better. However, our results are not conclusive in comparing the LSTM and the GRU, which suggests that the choice of the type of gated recurrent unit may depend heavily on the dataset and corresponding task."

[Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling](https://arxiv.org/abs/1412.3555) by Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio

"The GRU outperformed the LSTM on all tasks with the exception of language modelling"

[An Empirical Exploration of Recurrent Network Architectures](http://proceedings.mlr.press/v37/jozefowicz15.pdf) by Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever

"Our consistent finding is that depth of at least two is beneficial. However, between two and three layers our results are mixed. Additionally, the results are mixed between the LSTM and the GRU, but both significantly outperform the RNN."

[Visualizing and Understanding Recurrent Networks](https://arxiv.org/abs/1506.02078) by Andrej Karpathy, Justin Johnson, Li Fei-Fei

"Which of these variants is best? Do the differences matter? [Greff, et al. (2015)](https://arxiv.org/pdf/1503.04069.pdf) do a nice comparison of popular variants, finding that they’re all about the same. [Jozefowicz, et al. (2015)](http://proceedings.mlr.press/v37/jozefowicz15.pdf) tested more than ten thousand RNN architectures, finding some that worked better than LSTMs on certain tasks."

[Understanding LSTM Networks](https://colah.github.io/posts/2015-08-Understanding-LSTMs/) by Chris Olah

"In our [Neural Machine Translation] experiments, LSTM cells consistently outperformed GRU cells. Since the computational bottleneck in our architecture is the softmax operation we did not observe large difference in training speed between LSTM and GRU cells. Somewhat to our surprise, we found that the vanilla decoder is unable to learn nearly as well as the gated variant."

[Massive Exploration of Neural Machine Translation Architectures](https://arxiv.org/abs/1703.03906v2) by Denny Britz, Anna Goldie, Minh-Thang Luong, Quoc Le

**Example RNN Architectures**

| **Application** | **Cell** | **Layers** | **Size** | **Vocabulary** | **Embedding Size** | **Learning Rate** |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Speech Recognition (large vocabulary) | LSTM | 5, 7 | 600, 1000 | 82K, 500K | -- | -- | [paper](https://arxiv.org/abs/1610.09975) |
| Speech Recognition | LSTM | 1, 3, 5 | 250 | -- | -- | 0.001 | [paper](https://arxiv.org/abs/1303.5778) |
| Machine Translation (seq2seq) | LSTM | 4 | 1000 | Source: 160K, Target: 80K | 1,000 | -- | [paper](https://arxiv.org/abs/1409.3215) |
| Image Captioning | LSTM | -- | 512 | -- | 512 | (fixed) | [paper](https://arxiv.org/abs/1411.4555) |
| Image Generation | LSTM | -- | 256, 400, 800 | -- | -- | -- | [paper](https://arxiv.org/abs/1502.04623) |
| Question Answering | LSTM | 2 | 500 | -- | 300 | -- | [pdf](http://www.aclweb.org/anthology/P15-2116) |
| Text Summarization | GRU |  | 200 | Source: 119K, Target: 68K | 100 | 0.001 | [pdf](https://pdfs.semanticscholar.org/3fbc/45152f20403266b02c4c2adab26fb367522d.pdf) |

If you want to learn more about hyperparameters, these are some great resources on the topic:

* [Practical recommendations for gradient-based training of deep architectures](https://arxiv.org/abs/1206.5533) by Yoshua Bengio
* [Deep Learning book - chapter 11.4: Selecting Hyperparameters](http://www.deeplearningbook.org/contents/guidelines.html) by Ian Goodfellow, Yoshua Bengio, Aaron Courville
* [Neural Networks and Deep Learning book - Chapter 3: How to choose a neural network's hyper-parameters?](http://neuralnetworksanddeeplearning.com/chap3.html#how_to_choose_a_neural_network's_hyper-parameters) by Michael Nielsen
* [Efficient BackProp (pdf)](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf) by Yann LeCun

More specialized sources:

* [How to Generate a Good Word Embedding?](https://arxiv.org/abs/1507.05523) by Siwei Lai, Kang Liu, Liheng Xu, Jun Zhao
* [Systematic evaluation of CNN advances on the ImageNet](https://arxiv.org/abs/1606.02228) by Dmytro Mishkin, Nikolay Sergievskiy, Jiri Matas
* [Visualizing and Understanding Recurrent Networks](https://arxiv.org/abs/1506.02078) by Andrej Karpathy, Justin Johnson, Li Fei-Fei