

From data to a deep learning model: Deep learning workflow / Recipe

The universal workflow of machine learning

Step 1: Define the problem and assemble a dataset ✓

- Binary / multi-class classification or regression?

Step 2: Choose a measure of success ✓

- Accuracy (for balanced), Precision / Recall (for unbalanced)

Step 3: Decide on an evaluation protocol ✓

- Holdout / K-fold cross validation

Step 4: Prepare your data ✓

- Normalization / standardization

Step 5: Develop a model that does better than a baseline ✓

Step 6: Scale up: Develop a model that overfits ✓

Step 7: Regularize your model and tune hyperparameters ✓

Step 5: Develop a model that does better than the baseline

- Can you achieve "statistical power"?
 - a small model that is capable of beating a dumb baseline
- Example 1:
 - In the MNIST digit-classification example, anything that achieves an accuracy greater than 0.1
- Example 2:
 - In the IMDB example, it's anything with an accuracy greater than 0.5

Step 5: Develop a model that does better than the baseline

- Can you achieve "statistical power"?
 - a small model that is capable of beating a dumb baseline
- Example 1:
 - In the MNIST digit-classification example, anything that achieves an accuracy greater than 0.1
- Example 2:
 - In the IMDB example, it's anything with an accuracy greater than 0.5
- What can you do, if you cannot build a model with statistical power?
 - Check the last-layer's activation are you using sigmoid for regression?
 - Check the loss function binary_crossentropy for classification and mse/mae for regression
 - Check optimizer start with rmsprop with its default learning rate
 - Last resort: Use output as one of the input features

Step 6: Scale up: Develop a model that overfits

- So, we obtained a model that has statistical power, what next?
- Is our model sufficiently powerful to learn more patterns?
 - Does it have enough layers and parameters to properly model the problem at hand?

Step 6: Scale up: Develop a model that overfits

- So, we obtained a model that has statistical power, what next?
- Is our model sufficiently powerful to learn more patterns?
 - Does it have enough layers and parameters to properly model the problem at hand?
- To figure out if you can overfit, you have to overfit
 - How to overfit?
 - Add layers, Make the layers bigger, and Train for more epochs
 - Monitor the training loss and validation loss (and accuracy/mae)
 - When you see that the model's performance on the validation data begins to degrade, you've achieved overfitting

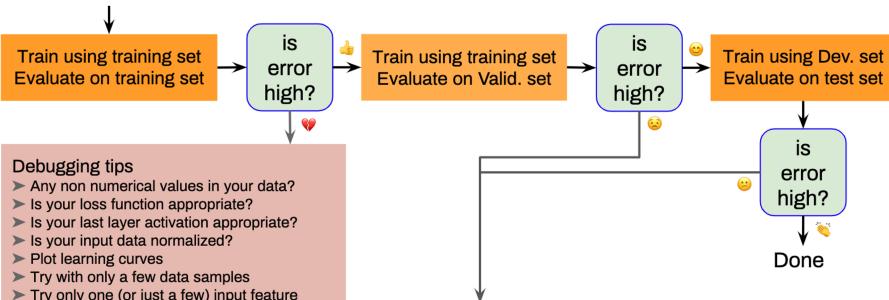
Step 7: Regularize the model and tune hyper-parameters

- Repeatedly modify your model, train it, evaluate on your validation data
 - Repeat, until the model is as good as it can get
- What to change?
 - Add dropout
 - Try different architectures: add or remove layers
 - Add L1 and/or L2 regularization
 - Try different hyperparameters (such as the number of units per layer or the learning rate of the optimizer) to find the optimal configuration
 - [Optional] Add new features or remove features that don't seem to be informative

Step 7: Regularize the model and tune hyper-parameters

- Repeatedly modify your model, train it, evaluate on your validation data
 - Repeat, until the model is as good as it can get
- What to change?
 - Add dropout
 - Try different architectures: add or remove layers
 - Add L1 and/or L2 regularization
 - Try different hyperparameters (such as the number of units per layer or the learning rate of the optimizer) to find the optimal configuration
 - [Optional] Add new features or remove features that don't seem to be informative
- Once you've developed a satisfactory model configuration
 - Train your final production model on all the available data (training and validation) and evaluate it one last time on the test set

How to debug a deep learning development pipeline?



Further debugging

- ➤ Get more data (more real data)
- ➤ Try regularization techniques
- Try new model architectures
- ➤ Is your data shuffled?

Regularization techniques

- Data augmentation (more fake data)
- Reduce the network's size
- Add weight regularization (L1/L2)
- Add dropout lavers
- Early stopping & checkpointing
- Adding batch normalization layers

- Try only one (or just a few) input feature
- ➤ Does your model have too many layers?
- ➤ Do the output labels need to be normalized?
- Check what values your model is generating
- ➤ Check if the loss is decreasing over epochs
- > Add output labels as one of the input features
- Train bigger models (more neurons & layers)
- Train longer (more epochs)
- Try new model architectures

UMSL - Deep Learning ML workflow - Slide 9 Badri Adhikar