DTU Course 02456 Deep learning
4 Tricks of the trade
2020 Updates

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

October 4, 2020

Objectives of 2020 updates - More tricks of the trade!

- Andrej Karpathy blog post
- P1: Become one with the data
- P2: Simple baselines
- P3: Overfit, regularize, tune and tune some more
- Quiz

Part 1:

Not knowing your data is a recipe for failure

A Recipe for Training Neural Networks

- Large gap between

  here is how a convolutional layer works

  and

  our convnet achieves state of the art results

A Recipe for Training Neural Networks

- Large gap between

  here is how a convolutional layer works

  and

  our convnet achieves state of the art results
- Not talk about common errors but rather
- about how one can avoid making these errors altogether (or fix them very fast)

A Recipe for Training Neural Networks

- Large gap between

  here is how a convolutional layer works

  and

  our convnet achieves state of the art results
- Not talk about common errors but rather
- about how one can avoid making these errors altogether (or fix them very fast)
- Neural net training is a leaky abstraction

```
>>> your_data = # plug your awesome dataset here
>>> model = SuperCrossValidator(SuperDuper.fit, your_data, ResNet50, SGDOptimizer)
# conquer world here
```

- If you insist on using the technology without understanding how it works you are likely to fail.

A Recipe for Training Neural Networks

- Large gap between
  here is how a convolutional layer works
  and
  our convnet achieves state of the art results
- Not talk about common errors but rather
- about how one can avoid making these errors altogether (or fix them very fast)
- Neural net training is a leaky abstraction

```
>>> your_data = # plug your awesome dataset here
>>> model = SuperCrossValidator(SuperDuper.fit, your_data, ResNet50, SGDOptimizer)
# conquer world here
```

- If you insist on using the technology without understanding how it works you are likely to fail.
- Neural net training fails silently
- a “fast and furious” approach to training neural networks does not work (in 2020)

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?
- How much variation is there and what form does it take?

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?
- How much variation is there and what form does it take?
- What variation is spurious and could be preprocessed out?

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?
- How much variation is there and what form does it take?
- What variation is spurious and could be preprocessed out?
- Does spatial position matter or do we want to average pool it out?

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?
- How much variation is there and what form does it take?
- What variation is spurious and could be preprocessed out?
- Does spatial position matter or do we want to average pool it out?
- How much does detail matter and how far could we afford to downsample the images?

Become one with the data

- Spend copious amount of time (measured in units of hours) scanning through thousands of examples,
- understanding their distribution and looking for patterns.
- Might find: duplicate examples, corrupted images / labels.
- Look for data imbalances and biases.
- Think about your process for classifying the data
- Are very local features enough or do we need global context?
- How much variation is there and what form does it take?
- What variation is spurious and could be preprocessed out?
- Does spatial position matter or do we want to average pool it out?
- How much does detail matter and how far could we afford to downsample the images?
- How noisy are the labels?

Become one with the data - now using the model

- When you have a qualitative sense then write some simple code to search/filter/sort by, e.g.
- type of label, size of annotations, number of annotations, etc. and
- visualize their distributions and the outliers along any axis.
- The outliers especially almost always uncover some bugs in data quality or preprocessing.

Become one with the data - now using the model

- When you have a qualitative sense then write some simple code to search/filter/sort by, e.g.
- type of label, size of annotations, number of annotations, etc. and
- visualize their distributions and the outliers along any axis.
- The outliers especially almost always uncover some bugs in data quality or preprocessing.
- Eventually you can use trained model as a compressed/compiled version of your dataset
- Look at network (mis)predictions and understand where they might be coming from
- If model is not consistent w data then something is off

Part 2:
Simple baselines

Set up the end-to-end training/evaluation skeleton + get dumb baselines

- Use simple model, linear model or tiny conv net
- Things we do: train it, visualize the losses, accuracy, model predictions, and perform ablations

Set up the end-to-end training/evaluation skeleton + get dumb baselines

- Use simple model, linear model or tiny conv net
- Things we do: train it, visualize the losses, accuracy, model predictions, and perform ablations
- Tips and tricks:
    - fix random seed.
    - simplify. No regularization, data augmentation, etc.
    - add significant digits to your eval. Loss on entire test set
    - verify loss at init. E.g cross entropy is $\log C$, $C =$ number of classes
    - init well. E.g. Glorot.
    - human baseline.

Set up the end-to-end training/evaluation skeleton + get dumb baselines

- Tips and tricks continued:
  - input-indepent baseline. Set all inputs to zero and train.
  - overfit one batch.
  - verify decreasing training loss. Increase model capacity little by little.
  - visualize data just before the net
  - visualize prediction dynamics. Prediction on fixed test batch.
  - use backprop to chart dependencies. Gradients give you information about what depends on what in your network, which can be useful for debugging. E.g. set loss = $\sum_d x_{id}$ and calculatte gradient with respect to inputs X.
  - generalize a special case. E.g. start implementation with functions with loops and later vectorize

Use backprop to chart dependencies - details

1. Create a multi-batch input (x = torch.rand([4, 3, 224, 224])
2. Set your input to be differentiable (x.requires_grad = True)
3. Set your model into evaluation mode (model.eval()) to avoid batch norm
4. Run a forward pass (out = model(x))
5. Define the loss as depending on one of the inputs (for instance: loss = out[2].sum())
6. Run a backprop (loss.backward)
7. Verify that only x[2] has non-null gradients: assert (x.grad[i] == 0.).all() for i != 2 and (x.grad[2] != 0).any()

Credit: Blogpost comment by @Elow2709.

Part 3:

Overfit, regularize and tune in that order

Scale up while overfitting

- get more data.
- data augment.
- creative augmentation.

Scale up while overfitting

- get more data.
- data augment.
- creative augmentation.
- use pretrained network (for images ImageNet and NLP BERT)
- stick with supervised learning (= don't do unsupervised yourself)

Scale up while overfitting

- get more data.
- data augment.
- creative augmentation.
- use pretrained network (for images ImageNet and NLP BERT)
- stick with supervised learning (= don't do unsupervised yourself)
- smaller input dimensionality.
- smaller model size. E.g. pooling can save a lot parameters.
- decrease the batch size. (= more regularization)

Scale up while overfitting

- get more data.
- data augment.
- creative augmentation.
- use pretrained network (for images ImageNet and NLP BERT)
- stick with supervised learning (= don't do unsupervised yourself)
- smaller input dimensionality.
- smaller model size. E.g. pooling can save a lot parameters.
- decrease the batch size. (= more regularization)
- drop. Add dropout. Use dropout2d (spatial dropout) for ConvNets.
- weight decay. Increase the weight decay penalty.
- early stopping.

Scale up while overfitting

- get more data.
- data augment.
- creative augmentation.
- use pretrained network (for images ImageNet and NLP BERT)
- stick with supervised learning (= don't do unsupervised yourself)
- smaller input dimensionality.
- smaller model size. E.g. pooling can save a lot parameters.
- decrease the batch size. (= more regularization)
- drop. Add dropout. Use dropout2d (spatial dropout) for ConvNets.
- weight decay. Increase the weight decay penalty.
- early stopping.
- try a larger model.
- (visualize first layer weights -conv net)

Tune

- random over grid search. Focus on parameters that make a difference
- hyper-parameter optimization. Bayesian optimization

Squeeze out the juice

- ensembles.
- leave it training

Squeeze out the juice

- ensembles.
- leave it training
- If you make until here: You are a master!

Quiz

- Recap: why do we split our data into training, validation and tests sets?
- Data: What does Andrei Karpathy mean when he says that the model is compressed/compiled version of your dataset?
- Data: If we accept that why is it then a good idea to spend time really understanding the data?
- Data: What else might we find when we explore the data?
- Data: List a number of statistics we can compute on the data before starting modelling it.
- Baselines: What components should our training set-up have?
- Baselines: Implement chart dependies in Pytorch for one of the models we have use so far. Does it work as it should?
- Scaling up: How do see that a model is overfitting?
- Scaling up: Why should we overfit in this phase?