How To Improve Deep Learning Performance

by **Jason Brownlee** on September 21, 2016 in **Deep Learning**

20 Tips, Tricks and Techniques That You Can Use To Fight Overfitting and Get Better Generalization

How can you get better performance from your deep learning model?

It is one of the most common questions I get asked.

It might be asked as:

> *How can I improve accuracy?*

…or it may be reversed as:

> *What can I do if my neural network performs poorly?*

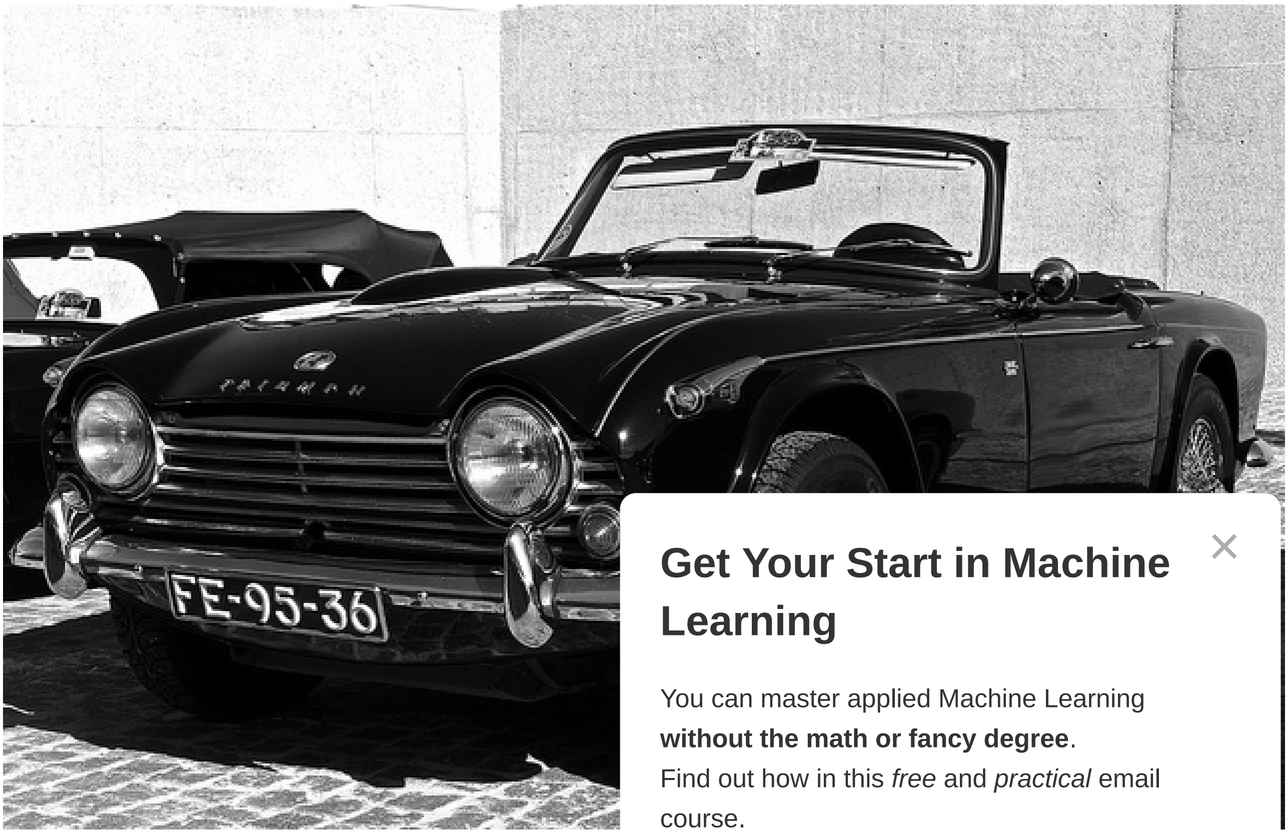
I often reply with "*I don't know exactly, but I have lots of ideas.*"

Then I proceed to list out all of the ideas I can think of that might give a lift in performance.

Rather than write out that list again, I've decided to put all of my ideas into this post.

The ideas won't just help you with deep learning, but really any machine learning algorithm.

It's a big post, you might want to bookmark it.



How To Improve Dee
Photo by Pedro Ribeiro S

Ideas to Improve Algorithm P

This list of ideas is not complete but it is a great start.

My goal is to give you lots ideas of things to try, hopefully, one or two ideas that you have not thought of.

You often only need one good idea to get a lift.

If you get results from one of the ideas, let me know in the comments.
I'd love to hear about it!

If you have one more idea or an extension of one of the ideas listed, let me know, I and all readers would benefit! It might just be the one idea that helps someone else get their breakthrough.

I have divided the list into 4 sub-topics:

1. ***Improve Performance With Data.***
2. ***Improve Performance With Algorithms.***
3. ***Improve Performance With Algorithm Tuning.***
4. ***Improve Performance With Ensembles.***

The gains often get smaller the further down the list. For example, a new framing of your problem or more data is often going to give you more payoff than tuning the parameters of your best performing algorithm. Not always, but in general.

I have included lots of links to tutorials from the blog, questions from related sites as well as questions on the classic Neural Net FAQ.

Some of the ideas are specific to artificial neural networks, but many are quite general. General enough that you could use them to spark ideas on improving your performance with other techniques.

Let's dive in.

1. Improve Performance With Data

You can get big wins with changes to your training data and problem definition. Perhaps even the biggest wins.

Here's a short list of what we'll cover:

1. Get More Data.
2. Invent More Data.
3. Rescale Your Data.
4. Transform Your Data.
5. Feature Selection.

1) Get More Data

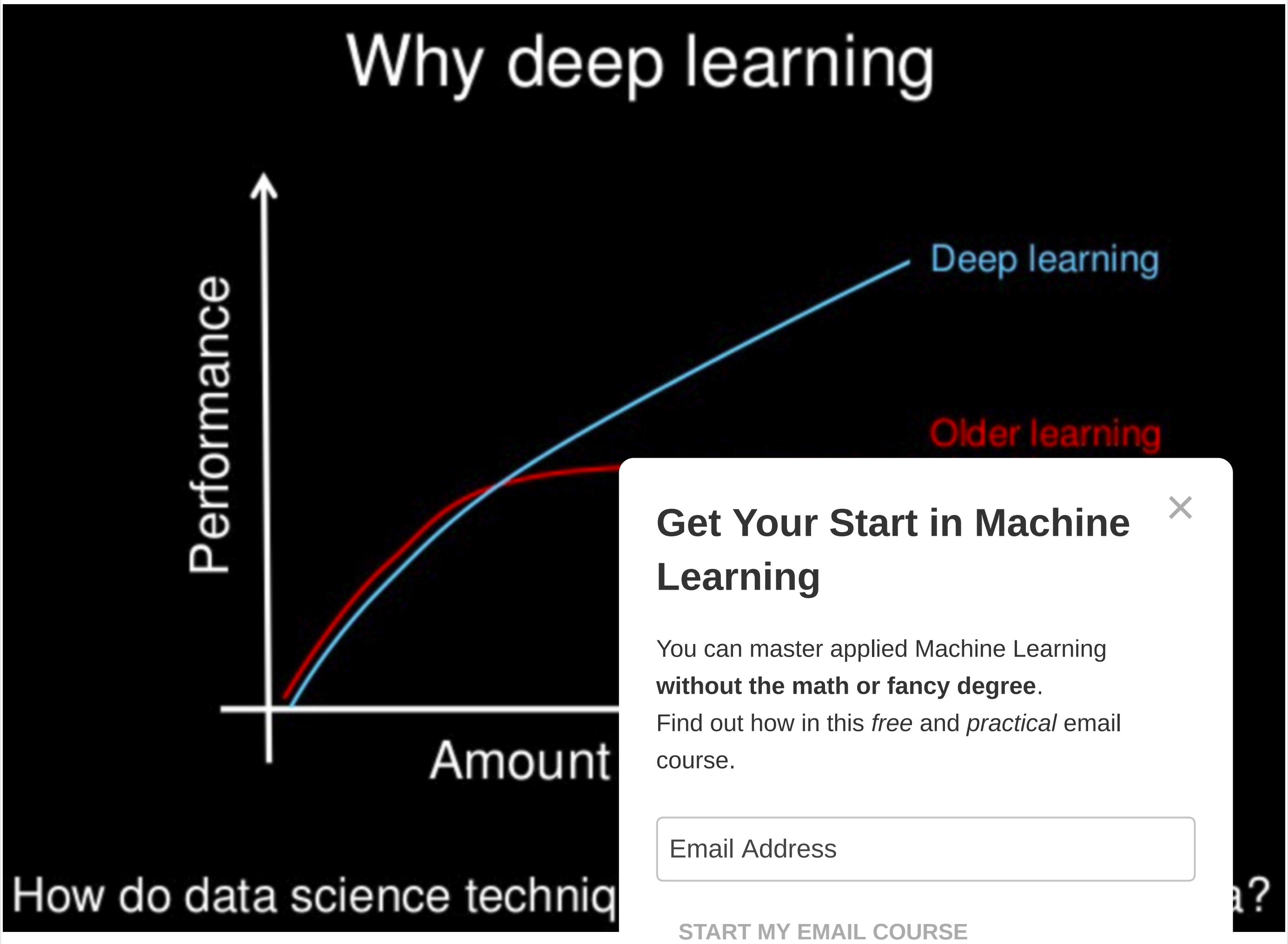
Can you get more training data?

The quality of your models is generally constrained
best data you can get for your problem.

You also want lots of it.

Deep learning and other modern nonlinear machine learning techniques get better with more data. Deep learning especially. It is one of the main points that make deep learning so exciting.

Take a look at the following cartoon:



Why De
Slide by Andrew Ng, all rights reserved.

More data does not always help, but it can. If I am given the choice, I will get more data for the optionality it provides.

Related:

- Datasets Over Algorithms

2) Invent More Data

Deep learning algorithms often perform better with more data.

We mentioned this in the last section.

If you can't reasonably get more data, you can invent more data.

- If your data are vectors of numbers, create randomly modified versions of existing vectors.
- If your data are images, create randomly modified versions of existing images.
- If your data are text, you get the idea…

Often this is called data augmentation or data generation.

You can use a generative model. You can also use simple tricks.

For example, with photograph image data, you can get big gains by randomly shifting and rotating existing images. It improves the generalization of the model to such transforms in the data if they are to be expected in new data.

This is also related to adding noise, what we used to call adding jitter. It can act like a regularization method to curb overfitting the training dataset.

Related:

- Image Augmentation for Deep Learning With Keras
- What is jitter? (Training with noise)

3) Rescale Your Data

This is a quick win.

A traditional rule of thumb when working with neur

Rescale your data to the bounds of your activation

If you are using sigmoid activation functions, resca using the Hyperbolic Tangent (tanh), rescale to val

This applies to inputs (x) and outputs (y). For exan predict binary values, normalize your y values to b benefit from normalizing your y values.

This is still a good rule of thumb, but I would go further.

I would suggest that you create a few different versions of your training dataset as follows:

- Normalized to 0 to 1.
- Rescaled to -1 to 1.
- Standardized.

Then evaluate the performance of your model on each. Pick one, then double down.

If you change your activation functions, repeat this little experiment.

Big values accumulating in your network are not good. In addition, there are other methods for keeping numbers small in your network such as normalizing activation and weights, but we'll look at these techniques later.

Related:

- Should I standardize the input variables (column vectors)?
- How To Prepare Your Data For Machine Learning in Python with Scikit-Learn

4) Transform Your Data

Related to rescaling suggested above, but more w

You must really get to know your data. Visualize it. Look for outliers.

Guesstimate the univariate distribution of each column.

- Does a column look like a skewed Gaussian, consider adjusting the skew with a Box-Cox transform.
- Does a column look like an exponential distribution, consider a log transform.
- Does a column look like it has some features, but they are being clobbered by something obvious, try squaring, or square-rooting.
- Can you make a feature discrete or binned in some way to better emphasize some feature.

Lean on your intuition. Try things.

- Can you pre-process data with a projection m
- Can you aggregate multiple attributes into a si
- Can you expose some interesting aspect of th
- Can you explore temporal or other structure ir

Neural nets perform feature learning. They can do

But they will also learn a problem much faster if yo o
the network for learning.

Spot-check lots of different transforms of your data
what doesn't.

Related:

- How to Define Your Machine Learning Problem
- Discover Feature Engineering, How to Engineer Features and How to Get Good at It
- How To Prepare Your Data For Machine Learning in Python with Scikit-Learn

5) Feature Selection

Neural nets are generally robust to unrelated data.

They'll use a near-zero weight and sideline the contribution of non-predictive attributes.

Still, that's data, weights, training cycles used on data not needed to make good predictions.

Can you remove some attributes from your data?

There are lots of feature selection methods and feature importance methods that can give you ideas of features to keep and features to boot.

Try some. Try them all. The idea is to get ideas.

Again, if you have time, I would suggest evaluating a few different selected "Views" of your problem with the same network and see how they perform.

- Maybe you can do as well or better with fewer

- Maybe all the feature selection methods boot the same specific subset of features. Yay, consensus on useless features.
- Maybe a selected subset gives you some ideas on further feature engineering you can perform. Yay, more ideas.

Related:

- An Introduction to Feature Selection
- Feature Selection For Machine Learning in Python

6) Reframe Your Problem

Step back from your problem.

Are the observations that you've collected the only

Maybe there are other ways. Maybe other framing structure of your problem to learning.

I really like this exercise because it forces you to o ted (ego!!!, time, money) in the current approach.

Even if you just list off 3-to-5 alternate framings an confidence in the chosen approach.

- Maybe you can incorporate temporal elements s.
- Maybe your classification problem can become a regression problem, or the reverse.
- Maybe your binary output can become a softmax output?
- Maybe you can model a sub-problem instead.

It is a good idea to think through the problem and it's possible framings before you pick up the tool, because you're less invested in solutions.

Nevertheless, if you're stuck, this one simple exercise can deliver a spring of ideas.

Also, you don't have to throw away any of your prior work. See the ensembles section later on.

Related:

- How to Define Your Machine Learning Problem

2. Improve Performance With Algorithms

Machine learning is about algorithms.

All the theory and math describes different approaches to learn a decision process from data (if we constrain ourselves to predictive modeling).

You've chosen deep learning for your problem. Is it really the best technique you could have chosen?

In this section, we'll touch on just a few ideas around algorithm selection before next diving into the specifics of getting the most from your chosen deep learning method.

Here's the short list

1. Spot-Check Algorithms.
2. Steal From Literature.
3. Resampling Methods.

Let's get into it.

1) Spot-Check Algorithms

Brace yourself.

You cannot know which algorithm will perform best

If you knew, you probably would not need machine

What evidence have you collected that your chose

Let's flip this conundrum.

No single algorithm can perform better than any ot possible problems. All algorithms are equal. This is theorem.

Maybe your chosen algorithms is not the best for your problem.

Now, we are not trying to solve all possible problems, but the new hotness in algorithm land may not be the best choice on your specific dataset.

My advice is to collect evidence. Entertain the idea that there are other good algorithms and given them a fair shot on your problem.

Spot-check a suite of top methods and see which fair well and which do not.

- Evaluate some linear methods like logistic regression and linear discriminate analysis.
- Evaluate some tree methods like CART, Random Forest and Gradient Boosting.
- Evaluate some instance methods like SVM and kNN.
- Evaluate some other neural network methods like LVQ, MLP, CNN, LSTM, hybrids, etc.

Double down on the top performers and improve their chance with some further tuning or data preparation.

Rank the results against your chosen deep learning method, how do they compare?

Maybe you can drop the deep learning model and use something a lot simpler, a lot faster to train, even something that is easy to understand.

Related:

- A Data-Driven Approach to Machine Learning
- Why you should be Spot-Checking Algorithms on your Machine Learning Problems
- Spot-Check Classification Machine Learning Algorithms in Python with scikit-learn

2) Steal From Literature

A great shortcut to picking a good method, is to steal ideas from literature.

Who else has worked on a problem like yours and what methods did they use.

Check papers, books, blog posts, Q&A sites, tutorials, everything Google throws at you.

Write down all the ideas and work your way throug

This is not about replicating research, it is about ne 'e
you a lift in performance.

Published researc

There are a lot of smart people writing lots of intere s
you need.

Related:

- How to Research a Machine Learning Algorit
- Google Scholar

3) Resampling Methods

You must know how good your models are.

Is your estimate of the performance of your models reliable?

Deep learning methods are slow to train.

This often means we cannot use gold standard methods to estimate the performance of the model such as k-fold cross validation.

- Maybe you are using a simple train/test split, this is very common. If so, you need to ensure that the split is representative of the problem. Univariate stats and visualization are a good start.
- Maybe you can exploit hardware to improve the estimates. For example, if you have a cluster or an Amazon Web Services account, we can train *n*-models in parallel then take the mean and standard deviation of the results to get a more robust estimate.
- Maybe you can use a validation hold out set to get an idea of the performance of the model as it trains (useful for early stopping, see later).
- Maybe you can hold back a completely blind validation set that you use only after you have performed model selection.

Going the other way, maybe you can make the dataset smaller and use stronger resampling methods.

- Maybe you see a strong correlation with the performance of the model trained on a sample of the training dataset as to one trained on the whole dataset. Perhaps you can perform model selection and tuning using the smaller dataset, then scale the final technique up to the full dataset at the end.
- Maybe you can constrain the dataset anyway, take a sample and use that for all model development.

You must have complete confidence in the performance estimates of your models.

Related:

- Evaluate the Performance Of Deep Learning Models in Keras
- Evaluate the Performance of Machine Learning Algorithms in Python using Resampling

3. Improve Performance With

This is where the meat is.

You can often unearth one or two well-performing a
most from those algorithms can take, days, weeks

Here are some ideas on tuning your neural networ

1. Diagnostics.
2. Weight Initialization.
3. Learning Rate.
4. Activation Functions.
5. Network Topology.
6. Batches and Epochs.
7. Regularization.
8. Optimization and Loss.
9. Early Stopping.

You may need to train a given “configuration” of your network many times (3-10 or more) to get a good estimate of the performance of the configuration. This probably applies to all the aspects that you can tune in this section.

For a good post on hyperparameter optimization see:

- How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras

1) Diagnostics

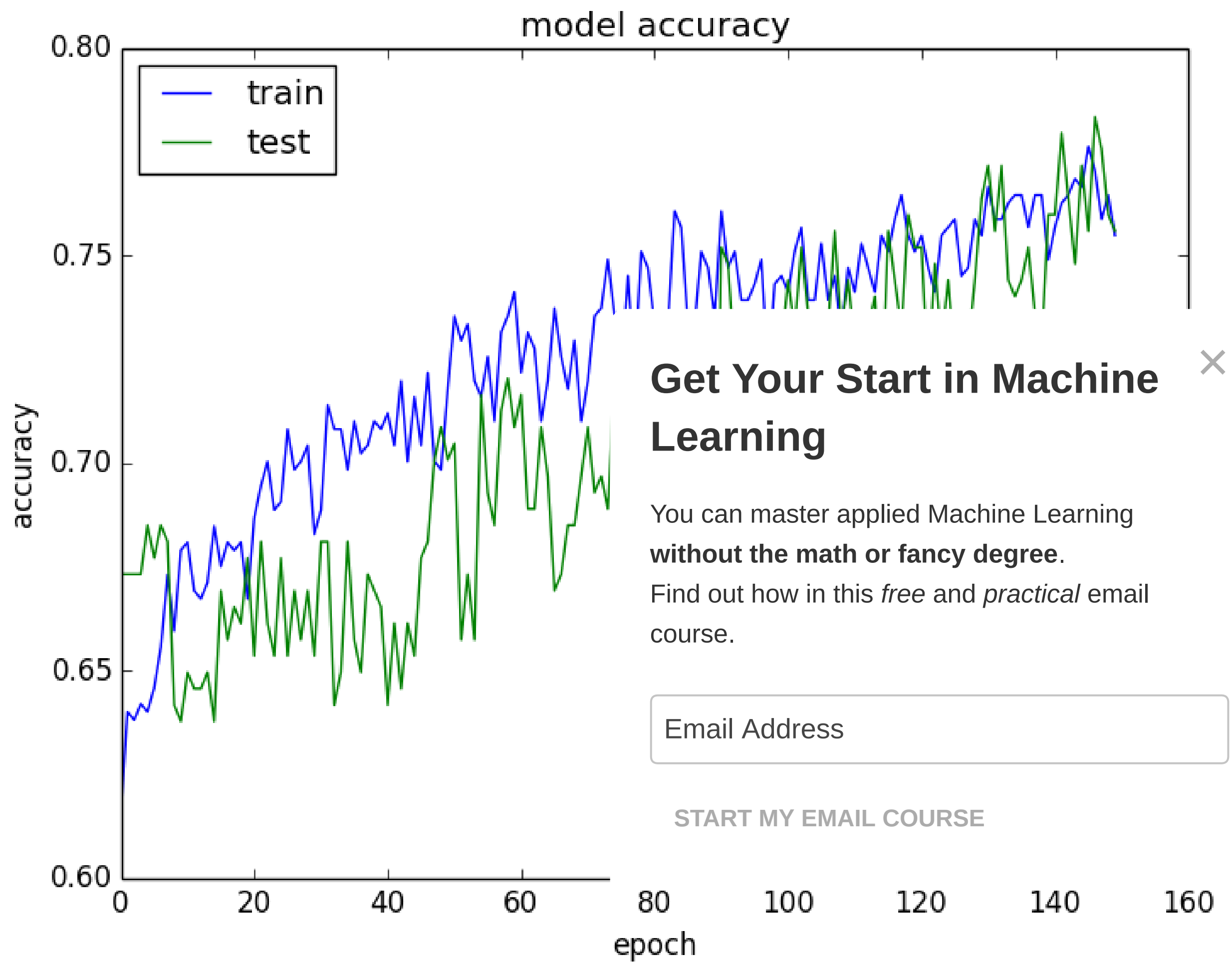
You will get better performance if you know why performance is no longer improving.

Is your model overfitting or underfitting?

Always keep this question in mind. Always.

It will be doing one or the other, just by varying degrees.

A quick way to get insight into the learning behavior of your model is to evaluate it on the training and a validation dataset each epoch, and plot the results.



Plot of Model Accuracy on Train and Validation Datasets

- If training is much better than the validation set, you are probably overfitting and you can use techniques like regularization.
- If training and validation are both low, you are probably underfitting and you can probably increase the capacity of your network and train more or longer.
- If there is an inflection point when training goes above the validation, you might be able to use early stopping.

Create these plots often and study them for insight into the different techniques you can use to improve performance.

These plots might be the most valuable diagnostics you can create.

Another useful diagnostic is to study the observations that the network gets right and wrong.

On some problems, this can give you ideas of things to try.

- Perhaps you need more or augmented examples of the difficult-to-train on examples.
- Perhaps you can remove large samples of the training dataset that are easy to model.
- Perhaps you can use specialized models that .

Related

- Display Deep Learning Model Training History in Keras
- Overfitting and Underfitting With Machine Learning Algorithms

2) Weight Initialization

The rule of thumb used to be:

> *Initialize using small random numbers.*

In practice, that is still probably good enough. But is it the best for your network?

There are also heuristics for different activation fur ... ıce in practice.

Keep your network fixed and try each initialization

Remember, the weights are the actual parameters ... e many sets of weights that give good performance,

- Try all the different initialization methods offer ... ınt.
- Try pre-learning with an unsupervised method
- Try taking an existing model and retraining a r ... fer learning)

Remember, changing the weight initialization method is closely tied with the activation function and even the optimization function.

Related

- Initialization of deep networks

3) Learning Rate

There is often payoff in tuning the learning rate.

Here are some ideas of things to explore:

- Experiment with very large and very small learning rates.
- Grid search common learning rate values from the literature and see how far you can push the network.
- Try a learning rate that decreases over epochs.
- Try a learning rate that drops every fixed number of epochs by a percentage.
- Try adding a momentum term then grid search learning rate and momentum together.

Larger networks need more training, and the reverse. If you add more neurons or more layers, increase your learning rate.

Learning rate is coupled with the number of trainin

Related:

- Using Learning Rate Schedules for Deep Learning Models in Python with Keras
- What learning rate should be used for backprop?

4) Activation Functions

You probably should be using rectifier activation functions.

They just work better.

Before that it was sigmoid and tanh, then a softmax, linear or sigmoid on the output layer. I don't recommend trying more than that unless you know what you're doing.

Try all three though and rescale your data to meet

Obviously, you want to choose the right transfer fu exploring different representations.

For example, switch your sigmoid for binary classi process your outputs. This may also require chang st- te. See the section on Data Transforms for more idea Related:

- Why use activation functions?

5) Network Topology

Changes to your network structure will pay off.

How many layers and how many neurons do you need?

No one knows. No one. Don't ask.

You must discover a good configuration for your problem. Experiment.

- Try one hidden layer with a lot of neurons (wide).
- Try a deep network with few neurons per layer (deep).
- Try combinations of the above.
- Try architectures from recent papers on problems similar to yours.
- Try topology patterns (fan out then in) and rules of thumb from books and papers (see links below).

It's hard. Larger networks have a greater representational capability, and maybe you need it.

More layers offer more opportunity for hierarchical re-composition of abstract features learned from the data. Maybe you need that.

Later networks need more training, both in epochs and in learning rate. Adjust accordingly.
Related:

These links will give you lots of ideas of things to t

- [How many hidden layers should I use?]
- [How many hidden units should I use?]

6) Batches and Epochs

The batch size defines the gradient and how often to update weights. An epoch is the entire training data exposed to the network, batch-by-batch.

Have you experimented with different batch sizes and number of epochs?

Above, we have commented on the relationship between learning rate, network size and epochs.

Small batch sizes with large epoch size and a larg ... rn deep learning implementations.

This may or may not hold with your problem. Gath

- Try batch size equal to training data size, men
- Try a batch size of one (online learning).
- Try a grid search of different mini-batch sizes
- Try training for a few epochs and for a heck of

Consider a near infinite number of epochs and set model seen so far, see more on this further down.

Some network architectures are more sensitive tha ... ns as often robust to batch size, whereas LSTM and CNNs quite sensitive, but that is just anecdotal.

Related

- What are batch, incremental, on-line … learning?
- Intuitively, how does mini-batch size affect the performance of (stochastic) gradient descent?

7) Regularization

Regularization is a great approach to curb overfitting the training data.

The hot new regularization technique is dropout, have you tried it?

Dropout randomly skips neurons during training, forcing others in the layer to pick up the slack. Simple and effective. Start with dropout.

- Grid search different dropout percentages.
- Experiment with dropout in the input, hidden and output layers.

There are extensions on the dropout idea that you can also play with like drop connect.

Also consider other more traditional neural network regularization techniques , such as:

- Weight decay to penalize large weights.
- Activation constraint, to penalize large activati

Experiment with the different aspects that can be penalized and with the different types of penalties that can be applied (L1, L2, both).

Related:

- Dropout Regularization in Deep Learning Models With Keras
- What is Weight Decay?

8) Optimization and Loss

It used to be stochastic gradient descent, but now there are a ton of optimizers.

Have you experimented with different optimization procedures?

Stochastic Gradient Descent is the default. Get the
momentum and learning rate schedules.

Many of the more advanced optimization methods ... er
convergence. This is good and bad, depending on

To get the most out of a given method, you really r ... hen
grid search different values for your problem. Hard

I have found that newer/popular methods can conv
capability of a given network topology, for example

- ADAM
- RMSprop

You can also explore other optimization algorithms such as the more traditional (Levenberg-Marquardt) and the less so (genetic algorithms). Other methods can offer good starting places for SGD and friends to refine.

The loss function to be optimized might be tightly related to the problem you are trying to solve.

Nevertheless, you often have some leeway (MSE and MAE for regression, etc.) and you might get a small bump by swapping out the loss function on your problem. This too may be related to the scale of your input data and activation functions that are being used.

Related:

- An overview of gradient descent optimization algorithms
- What are conjugate gradients, Levenberg-Marquardt, etc.?
- On Optimization Methods for Deep Learning, 2011 [PDF]

9) Early Stopping

You can stop learning once performance starts to degrade.

This can save a lot of time, and may even allow yo
evaluate the performance of your model.

Early stopping is a type of regularization to curb overfitting of the training data and requires that you monitor the performance of the model on training and a held validation datasets, each epoch.

Once performance on the validation dataset starts to degrade, training can stop.

You can also setup checkpoints to save the model if this condition is met (measuring loss of accuracy), and allow the model to keep learning.

Checkpointing allows you to do early stopping without the stopping, giving you a few models to choose from at the end of a run.

Related:

- How to Check-Point Deep Learning Models in
- What is early stopping?

4. Improve Performance With

You can combine the predictions from multiple mo

After algorithm tuning, this is the next big area for i

In fact, you can often get good performance from c enough" models rather than from multiple highly tu

We'll take a look at three general areas of ensemb

1. Combine Models.
2. Combine Views.
3. Stacking.

1) Combine Models

Don't select a model, combine them.

If you have multiple different deep learning models, each that performs well on the problem, combine their predictions by taking the mean.

The more different the models, the better. For example, you could use very different network topologies or different techniques.

The ensemble prediction will be more robust if each model is skillful but in different ways.

Alternately, you can experiment with the converse position.

Each time you train the network, you initialize it with different weights and it converges to a different set of final weights. Repeat this process many times to create many networks, then combine the predictions of these networks.

Their predictions will be highly correlated, but it might give you a small bump on those patterns that are harder to predict.

Related:

- Ensemble Machine Learning Algorithms in Python with scikit-learn
- How to Improve Machine Learning Results

2) Combine Views

As above, but train each network on a different view or framing of your problem.

Again, the objective is to have models that are skillf... predictions).

You can lean on the very different scaling and tran... for ideas.

The more different the transforms and framing of tl... more likely your results will improve.

Using a simple mean of predictions would be a go...

3) Stacking

You can also learn how to best combine the predic...

This is called stacked generalization or stacking for short.

Often you can get better results over that of a mean of the predictions using simple linear methods like regularized regression that learns how to weight the predictions from different models.

Baseline reuslts using the mean of the predictions from the submodels, but lift performance with learned weightings of the models.

- Stacked Generalization (Stacking)

Conclusions

You made it.

Additional Resources

There's a lot of good resources, but few tie all the ideas together.

I'll list some resources and related posts that you may find interesting if you want to dive deeper.

- Neural Network FAQ
- How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras
- Must Know Tips/Tricks in Deep Neural Networ...
- How to increase validation accuracy with deep...

Know a good resource? Let me know, leave a comment.

Handle The Overwhelm

This is a big post and we've covered a lot of ground.

You do not need to do everything. You just need one good idea to get a lift in performance.

Here's how to handle the overwhelm:

1. Pick one group
    1. Data.
    2. Algorithms.
    3. Tuning.
    4. Ensembles.
2. Pick one method from the group.
3. Pick one thing to try of the chosen method.
4. Compare the results, keep if there was an imp
5. Repeat.

Share Your Results

Did you find this post useful?

Did you get that one idea or method that made a d

Let me know, leave a comment!
I would love to hear about it.

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional developer and a machine learning practitioner. He is dedicated to helping developers get started and get good at applied machine learning. Learn more.

View all posts by Jason Brownlee →

< Stochastic Gradient Boosting with XGBoost and scikit-lear

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

n >

64 Responses to *How To Improve De*

Xu Lu September 22, 2016 at 4:31 am #

Thank you for your advice!

Jason Brownlee September 22, 2016 at 5:31 am # REPLY

I'm glad you found it useful Xu Lu.

Nitin September 22, 2016 at 5:30 am # REPLY

It is really a comprehensive explanation, going to try it.

Jason Brownlee September 22, 2016 at 5:31 am # REPLY

Thanks Nitin, let me know how you go.

Dr. Sudhir Pathak September 22, 2016 at 12:13 pm # REPLY

Thank you for sharing – this is very useful

Get Your Start in Machine Learning

Jason Brownlee September 22, 2016 at 5:29 pm # REPLY

I'm glad to hear it!

Arifa September 22, 2016 at 3:14 pm # REPLY

What are mathematical challenges for Deep learning in Big data?

Jason Brownlee September 22, 2016 at

Great question.

Generally, deep learning is empirical. We don't even what the heck is going on inside.

How many layers? What learning rate? These analytically. For now.

A strong math theory could push back the emp

Daisuke September 25, 2016 at 10:43 pm # REPLY

Thank you for sharing great post, I really appreciate.
I think it is very helpful, so I'd like to share the idea with my Japanese followers.
So I'm making translated summary of this post.
Could I ask for permission to post my summary in http://qiita.com/daisukelab ?
(Japanese tech blog media)

And could I ask the detail for 2-3),
"Maybe you can constrain the dataset anyway, take a sample and use that for all model development."

Do you mean:
"If we use smaller subset of dataset, we could use the subset for completing model development to the end"?

Thank you!

Jason Brownlee September 26, 2016 at 6:59 am # REPLY

Please do not repost the material Daisuke.

My that comment I meant that working with a sample of your data, rather than all of the data has benefits – like increasing the speed of turning around models

Daisuke September 26, 2016 at 11:16 am # REPLY

Hi Jason,
OK I will not repost, though it is for spreading your idea with translation and lead people visit here.
And thank you for clarification!

Chintan zaveri September 30, 2016 at 7:42 am # REPLY

Amazing post.. Thanx for sharing informat... models

Jason Brownlee September 30, 2016 at

Thanks Chintan, I'm glad you found it

Fernando C. November 25, 2016 at 6:07 am #

This is my current favorite website!

I'm currently working on implementing some nlp for regressions and was wondering if I could improve my results. I'll try ensembles, as I have many models already trained.

Thanks!

Jason Brownlee November 25, 2016 at 9:34 am # REPLY

Best of luck Fernando, I'd love to hear how you go.

Lee Moon November 25, 2016 at 9:12 pm # REPLY

This is really good information what I found. Actually, I am working in Semantic Segmentation using Deep learning. It can view as an extension task of recognition task. As I read, I felt that all segmentation techniques have come from recognition (We can think that the recognition as encoding phase provides probability map, the segmentation task maps the probability maps to the image by using decode phase). Hence my opinion, I think that if any state-of-the-art recognition network architecture applies for segmentation task which can achieve more accuracy than segmentation using older recognition network architecture. Do you think so? What is good direction to improve segmentation accuracy? Thank you in advance

Alp ALTIN December 9, 2016 at 6:20 pm # REPLY

I am a newbie in deep learning and experimenting with existing examples, using the digits interface. Thank you for all the effort to simplify the topic, a technical documentation still well understandable for newcomers.
My interest is on detecting (and counting) particles via deep learning. After many many experiments with various samples, I realised particles too close to each (almost touching) other are counted as one, while there is a clear seperation, to my eye.

I am looking for an approach in how to handle this.

I tried some thresholding techniques on individual images in an image processing software and best results obtained with color thresholding. Although I have no idea whether thresholding is a part of particle detection in object detection processes by
integrate an equivalent process into object detectic
Regards,

Naoki January 21, 2017 at 5:15 pm #

thank you. I'll try some techniques of this

Jason Brownlee January 22, 2017 at 5:0

You're welcome Naoki, I'd love to hear about your results.

emma February 5, 2017 at 11:40 pm # REPLY

very comprehensive blog! Great work!

Jason Brownlee February 6, 2017 at 9:43 am # REPLY

Thanks emma, I hope it helps with your project.

Max S. February 17, 2017 at 9:26 am # REPLY

Hello,

I would like to know if there is an implementation in Keras of "drop connect".

Thanks you for your time

Jason Brownlee February 17, 2017 at 10:07 am # REPLY

I have not seen one Max, but I expect there will be something out there!

KERKENI L. March 10, 2017 at 3:28 am # REPLY

Hi Jason,

Thank you for sharing great post, I really appreciate.

you mentioned in section 2: Invent More Data

-If you can't reasonably get more data, you can inv

-If your data are vectors of numbers, create randor

—-> Have you an example how to create randomly

Thank you!

Jason Brownlee March 10, 2017 at 9:29

I don't but you could experiment with (

Some good ideas to try include:

– randomly replace a subset of values with randomly selected values in the data population
– add a small random value (select distribution to meet the data distribution for a column)

It is tricky, because you need the new data to be "reasonable" for the assigned class.

Consider a skim of the literature for more sophisticated methods.

Lucas April 7, 2017 at 8:46 am # REPLY

First of all thank you for the thorough explanation and rich material, it's been helping me quite a lot.

One thing that still troubles me is applying Levenberg-Marquardt in Python, more specifically in Keras. I'm dealing with non-ideal input variables to infer target and would like to go through a range of optimizers to test the network performance.
Since one of the best available in Matlab is Levenberg-Marquardt, it would very good (and provide comparison value between languages) if I could accurately apply it in keras to train my network.

I am currently using Keras with Theano backend. Any help at all would be appreciated.

Best,
Lucas

Jason Brownlee April 9, 2017 at 2:50 pm # REPLY

Here is an example of grid searching optimization algorithms:
http://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

Sorry, I do not have an example of the Levenberg-Marquardt algorithm in Python for Keras.

Marcin May 17, 2017 at 3:27 am # REPLY

Hi Jason,

Thank you for the great work and posts.

I have got a question- after improving deep learnin
75% in a binary classification problem. Using other
is when do i know that my model is the best possbi
extend my data has explanatory power?

Jason Brownlee May 17, 2017 at 8:39 a

Well done!

Great question. We can never know for sure, b

Generally, no, I'm not aware of methods to estimate explanatory power, I'm not even clear what that might mean.

Marcin May 17, 2017 at 7:14 pm # REPLY

What I meant by 'explanatory power', was the ability of data to distinguish that one record belongs to class 1, second to class 2, third again to class 1 and so on.
It is some kind of limitation of the dataset, that it can achieve max. accuracy for example at 80% and my question was is it a way to measure, that level.
Thank you for your answer, now i am ready to accept my model 🙂

Jason Brownlee May 18, 2017 at 8:35 am # REPLY

You can configure the model to output probabilities instead of classes, this may give the result you require.

Cyrus June 22, 2017 at 4:30 pm # REPLY

Finally! Come across all the techniques to improve your deep learning model in a nutshell!
Thanks a lot Jason!

Jason Brownlee June 23, 2017 at 6:40 am # REPLY

I hope that you find them useful Cyrus.

Parva June 27, 2017 at 11:00 pm # REPLY

Hi, I'm working on my final year project wh r videos (if possible). My problem is that i cannot finc me out with this problem by giving me some sugge about (150-200 gb) to make my algorithm more pre but there is no response from them.

Jason Brownlee June 28, 2017 at 6:26 a

Sorry, I don't know where to get such

Glendon July 24, 2017 at 12:35 pm # REPLY

Hey Jason, Do you know of any empirical evidence for the "Why Deep Learning?" slide by Andrew Ng. Specifically I am working on a text classification problem, I am finding BoW + (Linear SVM's or Logistic Regression) giving me the best performance (which is what I find in the literature at least pre 2015). I'm fairly new to Deep learning, I have been testing it out on my problem having seen stuff like https://arxiv.org/abs/1408.5882 perform well in the Rotten Tomatoes Kaggle text classification problem. Although there was some other fancy tricks that I think gave the CNN extra juice. Do you have any recommendations or any benchmarking studies in this area that demonstrate what Andrew Ng is claiming?

Jason Brownlee July 25, 2017 at 9:24 am # REPLY

Not specifically, sorry.

Nunu July 26, 2017 at 7:58 pm # REPLY

Dear Jason,
Thanks for this article 🙂 I have a question : how to calculate the total error of a network ?!

thanks in advance.
Nunu

Jason Brownlee July 27, 2017 at 7:59 am # REPLY

Evaluate it on test data and calculate an error score, such as RMSE for regression or Accuracy on classification.

Nunu July 27, 2017 at 10:49 pm # REPLY

ok I will try it. Thanks a lot

Jason Brownlee July 28, 2017 at 8:31 ar

Let me know what works for you.

Danny July 28, 2017 at 1:31 pm #

("Hyperbolic Tangent (tanh), rescale to val
WHOOOOPS!!, that might be important,now i know, lol!

Finalllllllllly! someone who has explain this wonderfully with structure, and not just said its a black box! great info Doc

I have a question! do you have any pointers for unbalanced data?

is it better to sacrifice other data to balance every class out?

Jason Brownlee July 29, 2017 at 8:01 am # REPLY

Thanks Danny.

Yes, see this post on imbalanced data:
http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/

Danny July 28, 2017 at 1:47 pm # REPLY

Examples of what I mean by unbalance data

=== self drive car ===
left = [0,1,0] =5k samples

right = [0,0,1]= 5k samples
forward = [1,0,0]=100k samples

===stocks====
Buy = [0,1,0] =5k samples
Sell = [0,0,1]= 5k samples
Hold = [1,0,0]=100k samples

etc……..

J September 6, 2017 at 10:03 am # REPLY

Undersampling, Oversampling, or Sm

diao July 28, 2017 at 6:01 pm #

great article

Jason Brownlee July 29, 2017 at 8:07 ar

Thanks!

K.D. August 7, 2017 at 8:41 pm # REPLY

Hi Jason,

A long try-and-test process led me to the same conclusions. So thank you very much! This post will serve for a lot of new comers to the keras/ deep learning area.

Question:
In a classic case, you normalize your data, you train the model and then you "de-normalize" (inverse using the scaler). Now, imagine that the model you are training is fed with its own output and the predicted outpùt is out of the scaler range, what would you do to improve the model's performance.
Thank you in advance for your feedback!

Kind Regards.

Jason Brownlee August 8, 2017 at 7:49 am # REPLY

Thanks, I'm glad it helped.

If you have data outside of the scalers range, you can force it in bounds or update the scaling.

I hope that helps.

Sam August 16, 2017 at 4:03 pm # REPLY

Hi Jason,

Thanks for sharing this valuable post

One thing I am still wondering about, I am interested to apply deep learning in data stream classification (real time prediction), but my concern is the execution time that the deep learning needs. Any idea how to speed it up or how to handle it for real time prediction.

Regards

Jason Brownlee August 16, 2017 at 5:01

For modestly sized data, the feed-forv
is very fast.

As for training the network in real-time, I would

Sam August 22, 2017 at 5:26 pm #

Many thanks Jason

My Nguyen August 23, 2017 at 6:16 am # REPLY

Been learning a lot from your posts. Thanks for posting. Just curious, what's up with the random pictures?:)

Jason Brownlee August 23, 2017 at 6:58 am # REPLY

Thanks!

I figure the pictures would lighten the mood, be something interesting to look at as we get deep into technical topics. I often choose pictures based on where I am going or want to go for a holiday, e.g. the beach, the forest, etc. Sometimes they are a pun (e.g. a pic of a gas pipeline for a pipeline method). Sometimes they are just random.

You're the first person in 4 years to ask about them 🙂

Devakar Kumar Verma August 24, 2017 at 2:28 pm # REPLY

Under “Rescale Your Data”, you have pointed about scaling and activation function. So if we scale the data between [-1,1], then we have to implicitly mention about activation function (i.e tanh function) in LSTM using Keras. Am i correct in assumption, or Keras will pass tanh activation function default in LSTM.

Jason Brownlee August 24, 2017 at 4:27 pm # REPLY

For LSTMs at the first hidden layer, you will want to scale your data to the range 0-1.

Devakar Kumar Verma August 2

Hi Jason,

In https://machinelearningmastery.com/tim python/ artcle , you have mentioned to sca not used any activation function.

In keras documentation, https://keras.io/lay linear or no activation (i.e a(x) =x).

So my question is whether i need to add in

Jason Brownlee August 25, 2017 at 6:42 am # REPLY

The default for the LSTM is sigmoid outputs with tanh used for internal gates. I would recommend scaling input data for LSTMs to between [0,1].

Some testing shows this results in better model skill, generally.

faruk ahmad September 30, 2017 at 3:01 am # REPLY

Thanks for this amazing writing. It is very useful.

Jason Brownlee September 30, 2017 at 7:46 am # REPLY

Thanks, I’m glad it is helping.

Robin October 24, 2017 at 11:27 pm # REPLY

A bit outdated but still very useful. Thanks for this great article!

Jason Brownlee October 25, 2017 at 6:47 am # REPLY

Thanks. What do you think it is missing Robin?

Nguyen Anh Nguyen October 30, 2017 at 10:16 pm # REPLY

Would you publish technique for "DNN/CNN incremental learning" please.

Thank you.

Jason Brownlee October 31, 2017 at 5:3

Thanks for the suggestion.

Leave a Reply

Name (required)

Email (will not be published) (required)

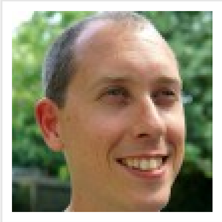
Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More



POPULAR

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
JULY 21, 2016

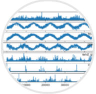
Your First Machine Learning Project in Python Step-By-Step
JUNE 10, 2016

Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016

How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

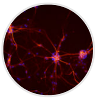
Multivariate Time Series Forecasting with LSTMs in Kera
AUGUST 14, 2017



Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016



How to Implement the Backpropagation Algorit
NOVEMBER 7, 2016

Get Your Start in Machine Learning

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Privacy | Contact | About

Get Your Start in Machine Learning