Hello ID24,

This is my first attempt after the first sneak peek of ID24 to tell a story about. To recap, this was the Youtube video you already know: https://www.youtube.com/watch?v=P2E8SMQPLcw
Without any doubt to me, you are the best explainer! And you might agree with me that ID24 could fill small books after this little video if we would talk about every single detail. So I will not even try to explain everything, but I would like to share the basic idea with you about ID24.

The work from 2023 is a short summary of parallel coordinates to distribution counts from 2023. Where ID24 is much further developed, but often consists of many ideas that have evolved since then. It was intended to show it for an ML job and also to show what we can do with it plus cool observations.

https://docs.google.com/document/d/1EBE3bdVu63zIGoBXTk_ibHa-X1Fmlt2Kdrkuh35jHZo/edit?usp=sharing

Another purpose of this work is, after 6 years as a daily reader of your blog, but without a degree or any other certification, to show what I have learned around ML.

ID24 seems like a new way of thinking like Reinforcement Learning with Human Feedback (RLHF), which was the way to ChatGPT. What seems to be a tribute from 2023 where most people today say AI, what ML was before. The AI button follows this hype, but it is actually a range based roulette wheel selection.

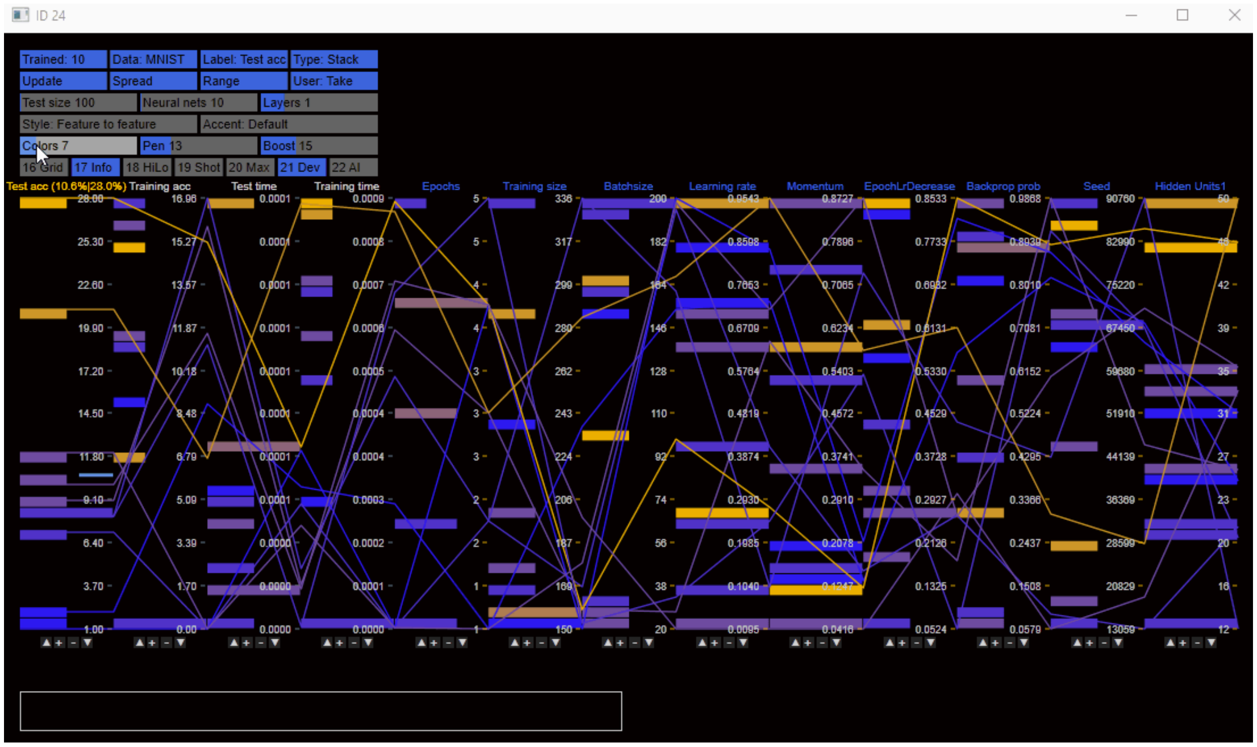
Back to ID24, it should be better seen as a game than a visualization, which might feel a bit unusual at first. The game is interchangeable and mouse based, which means we can take any data, put it in and play without typing. This is fast.

In the case of neural network hyperparameter optimization, we add the results of testing and training with accuracy and time. We could also add loss or other things. The next step is to add the hyperparameters to optimize the network or other systems, XGboost (https://xgboost.readthedocs.io/en/stable/parameter.html), Tsetlin Machines, Nuclear Plants or what comes next.

The paradigm now should be that everything we see is a feature for each column. The kicker is that we can use each feature as the label to color the connected data in lines and distribution counts as bars. This allows us to explore any relationship between the features from good to bad.

It is important to note the normalized view of the data. They are normalized for us visually, similar to how we would normalize data for neural networks. For dataset purposes, we should think of the features as a kind of glasses that gives us a connection to better understand the transfer of data to the system or the NN in this case.

It Seems to Work


ID 24

Trained: 10 | Data: MNIST | Label: Test acc | Type: Stack
Update | Spread | Range | User: Take
Test size 100 | Neural nets 10 | Layers 1
Style: Feature to feature | Accent: Default
Colors 7 | Pen 13 | Boost 15
16 Grid | 17 Info | 18 HiLo | 19 Shot | 20 Max | 21 Dev | 22 AI

One of the last steps and perhaps the most important was to find a way to see that everything was working, that the lines and bars were connected properly in every setting.

This is the activated developer mode. If you look carefully at the start of the animation on training accuracy and training time, you should see only 2 bars connected by 3 lines. Where the rest of the features show only bars with 1 or 2 counted lines.

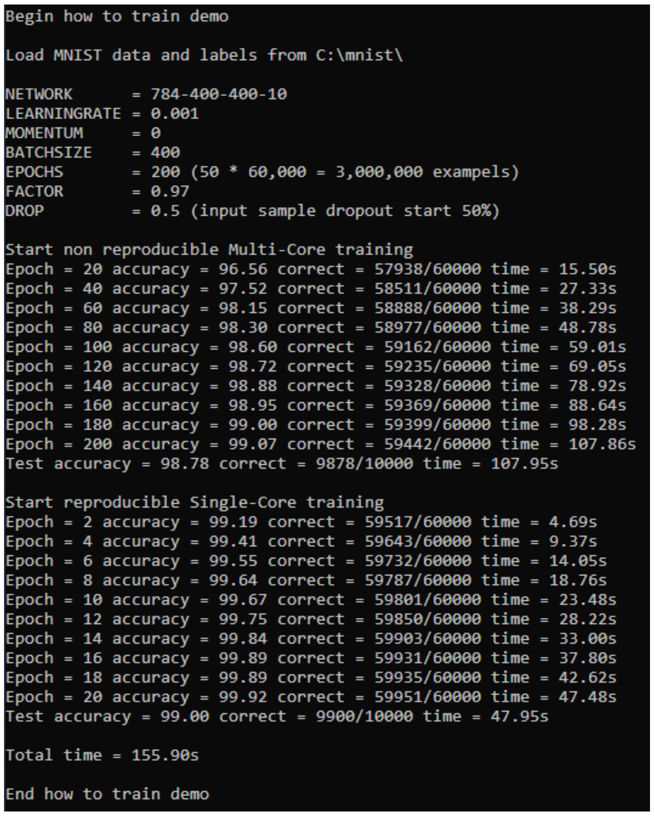
The Possibilities

Still a concept with a bit too much bling bling in there with distractions, but over 3 million visualization styles can be really useful in many ways. We are able to handle up to 10,000 examples as lines or more with ID24. However, with the distributions we are able to handle up to 20 million examples. And we can work with distributions to select a fraction of data to bring them into more understable lines.

The neural network is based on this implementation, which can predict more than 2 million handwritten digits per second on my $200 5600X 6-Cores CPU: https://github.com/grensen/neural_network_benchmark

In theory, this means that we can recognize one handwritten digit from every person in the world in an hour, while a CPU from 2024 should do it in 20 minutes. To get an idea from the scale.

The Expert to Set a System



```
Begin how to train demo

Load MNIST data and labels from C:\mnist\

NETWORK      = 784-400-400-10
LEARNINGRATE = 0.001
MOMENTUM     = 0
BATCHSIZE    = 400
EPOCHS       = 200 (50 * 60,000 = 3,000,000 exampels)
FACTOR       = 0.97
DROP         = 0.5 (input sample dropout start 50%)

Start non reproducible Multi-Core training
Epoch = 20 accuracy = 96.56 correct = 57938/60000 time = 15.50s
Epoch = 40 accuracy = 97.52 correct = 58511/60000 time = 27.33s
Epoch = 60 accuracy = 98.15 correct = 58888/60000 time = 38.29s
Epoch = 80 accuracy = 98.30 correct = 58977/60000 time = 48.78s
Epoch = 100 accuracy = 98.60 correct = 59162/60000 time = 59.01s
Epoch = 120 accuracy = 98.72 correct = 59235/60000 time = 69.05s
Epoch = 140 accuracy = 98.88 correct = 59328/60000 time = 78.92s
Epoch = 160 accuracy = 98.95 correct = 59369/60000 time = 88.64s
Epoch = 180 accuracy = 99.00 correct = 59399/60000 time = 98.28s
Epoch = 200 accuracy = 99.07 correct = 59442/60000 time = 107.86s
Test accuracy = 98.78 correct = 9878/10000 time = 107.95s

Start reproducible Single-Core training
Epoch = 2 accuracy = 99.19 correct = 59517/60000 time = 4.69s
Epoch = 4 accuracy = 99.41 correct = 59643/60000 time = 9.37s
Epoch = 6 accuracy = 99.55 correct = 59732/60000 time = 14.05s
Epoch = 8 accuracy = 99.64 correct = 59787/60000 time = 18.76s
Epoch = 10 accuracy = 99.67 correct = 59801/60000 time = 23.48s
Epoch = 12 accuracy = 99.75 correct = 59850/60000 time = 28.22s
Epoch = 14 accuracy = 99.84 correct = 59903/60000 time = 33.00s
Epoch = 16 accuracy = 99.89 correct = 59931/60000 time = 37.80s
Epoch = 18 accuracy = 99.89 correct = 59935/60000 time = 42.62s
Epoch = 20 accuracy = 99.92 correct = 59951/60000 time = 47.48s
Test accuracy = 99.00 correct = 9900/10000 time = 47.95s

Total time = 155.90s

End how to train demo
```

One nice neural net on MNIST reached 99% accuracy in the test, a great value in competition with other NN's: https://github.com/grensen/how_to_train

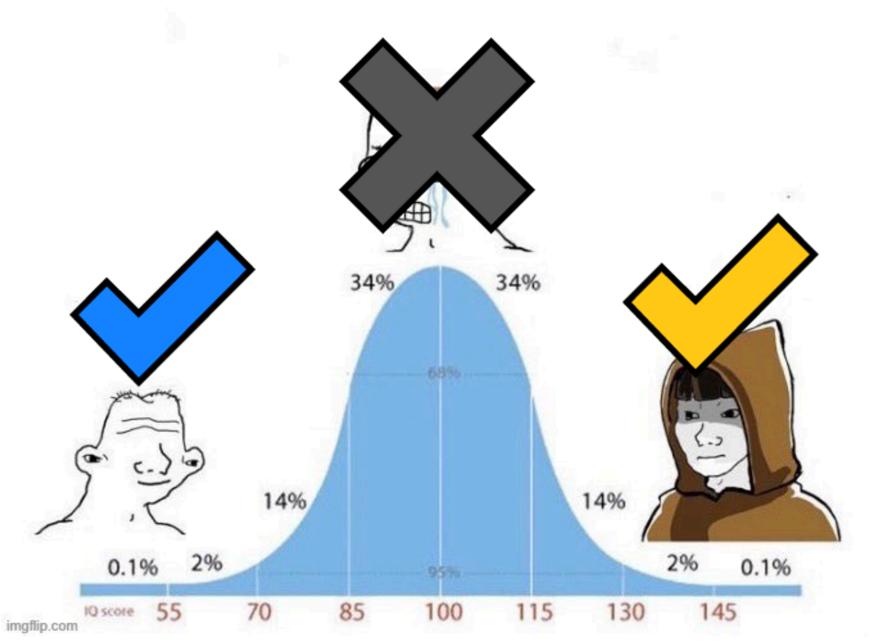
It took me a long time to get to this point, to understand the relationships between hyperparameters, training size, and so on. For example low learning rate and low momentum lead to ok-bad results. High learning rate and high momentum lead to the best results, but too high rates lead to the worst results. A beginner should be able to learn the same in 30 minutes with ID24 and enough computing power.

The common way was trial and error to gain experience that led to better and better results over the years. But how can I explain that these are good results and also work for other tasks, such as fraud detection?

Feeling is a bad teammate, but sometimes a teammate that we need. Sometimes a good feeling is all you need, then there is one more experiment.

For the moment, let us say there is already a good setting, but no experience. If the system needs to be changed for some reason, this could also affect the hyperparameters that need to be selected again. The better case seems to be to gain experience at the beginning which can act as an assistant that we can use to make further decisions if needed. But how?

Our Guide: All We Need to Know



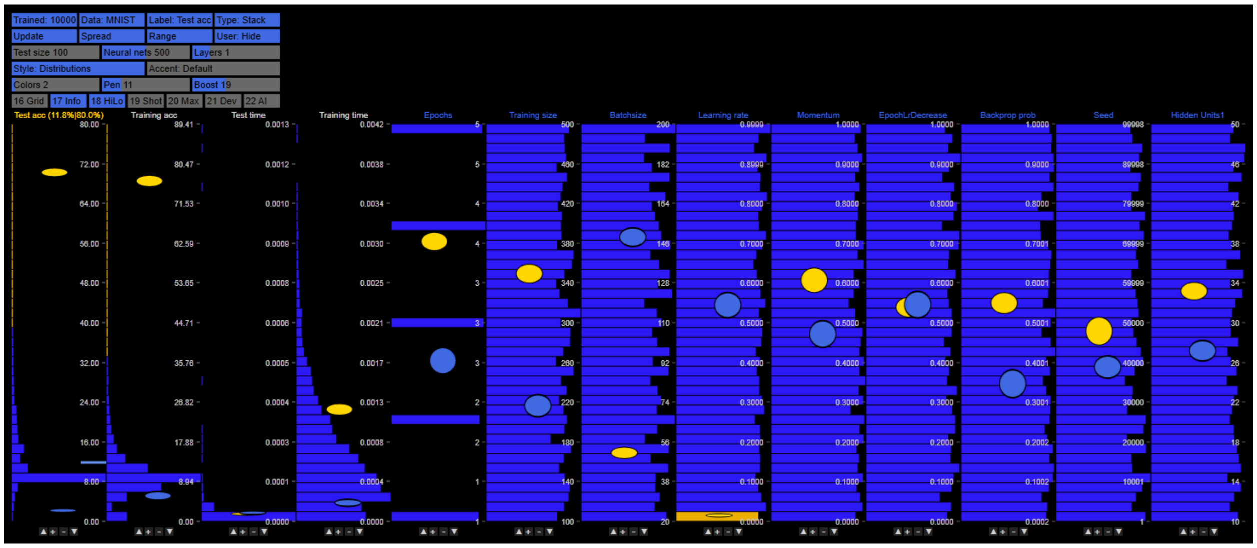
You may know this meme because it is the most popular meme in the Twitter AI/ML cosmos. But I think these memes are all wrong in some way. The takeaway might be that the dumbest person and the smartest person do the same thing.

I would argue that in reality there is a huge difference between the worst player and the best player. Learning what makes the best player different from the worst player will take the new player to a pretty good level. And to avoid bad surprises, we need to know both.

It turns out that we will make pretty good decisions by avoiding the left side, represented in blue, following the right side, represented in gold, and just ignore the noise in the middle. That's all, when we are at this point of the normal distribution.

The problem is the count, the normal distribution is not a single case, it is the normal distribution of many cases, where more seems to give at least a better picture. The normal can be very expensive.

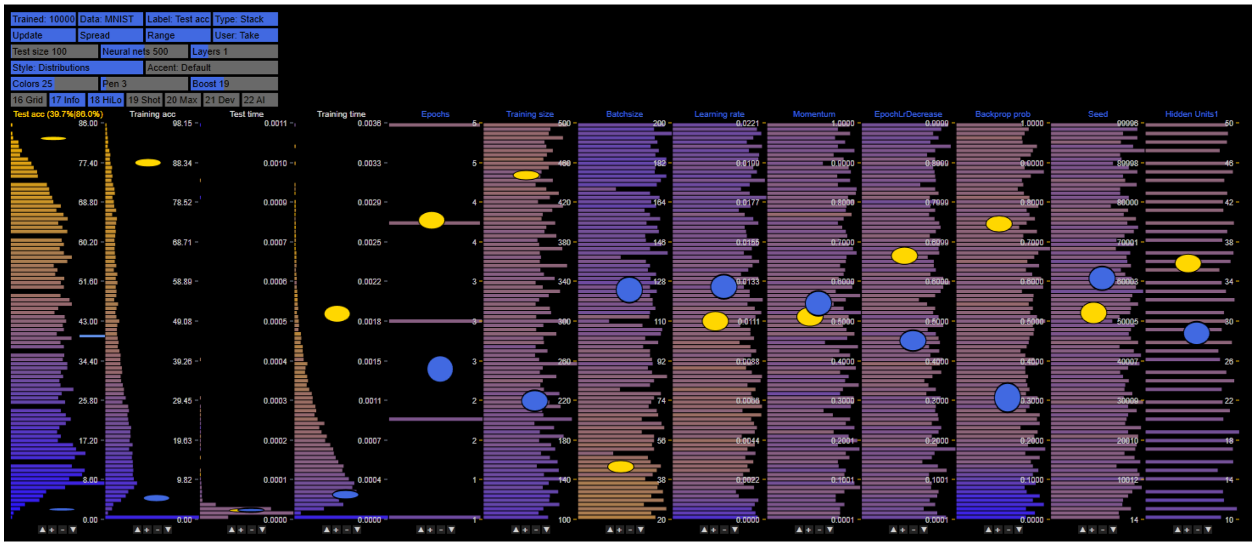
The First Step


Trained: 10000 | Data: MNIST | Label: Test acc | Type: Stack
Update | Spread | Range | User: Hide
Test size 100 | Neural nets 500 | Layers 1
Style: Distributions | Accent: Default
Colors 2 | Pen 11 | Boost 19
16 Grid | 17 Info | 18 HiLo | 19 Shot | 20 Max | 21 Dev | 22 AI

The next picture shows 10k trained NNs as distributions in 2 colors, the bar or bin size, showed the first gold for the learning rate. The balls can be activated with the HiLo button and represent the worst and best results after each update.

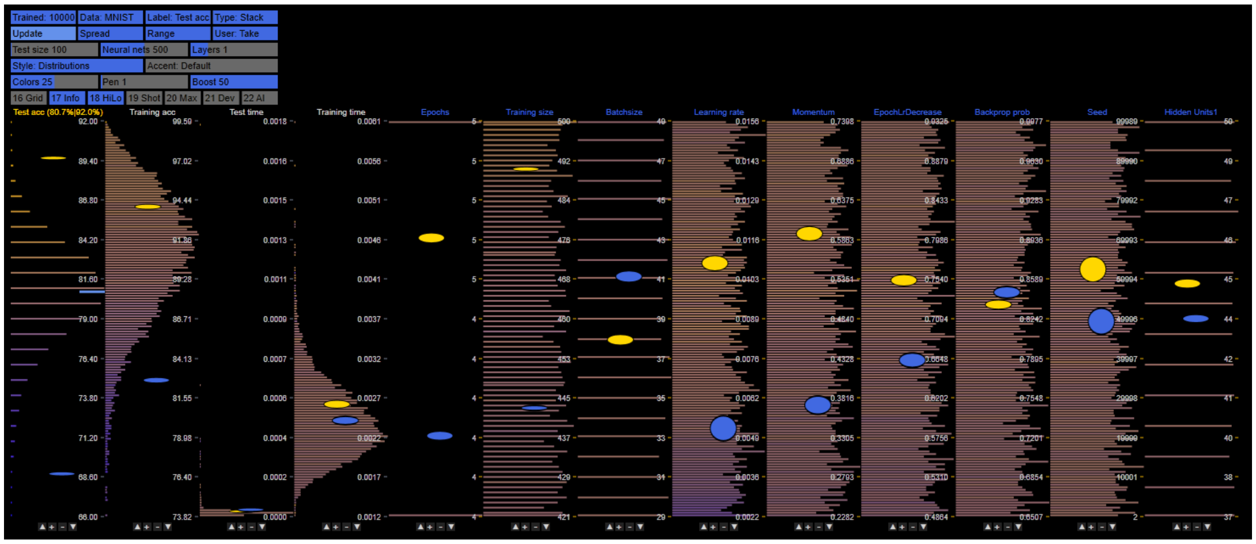
20 updates as mean is shown for the y-position and the variance for the height. This can help to make better decisions, even in this naive way. I manipulated the colors here to 2 to make a clear separation between the good and the bad half, then I changed the pen to the right bar size after the first yellow bar appeared: https://twitter.com/KleppeThorsten/status/1750136224684077534

The Update


Trained: 10000 | Data: MNIST | Label: Test acc | Type: Stack
Update | Spread | Range | User: Take
Test size 100 | Neural nets 500 | Layers 1
Style: Distributions | Accent: Default
Colors 25 | Pen 3 | Boost 19
16 Grid | 17 Info | 18 HiLo | 19 Shot | 20 Max | 21 Dev | 22 AI

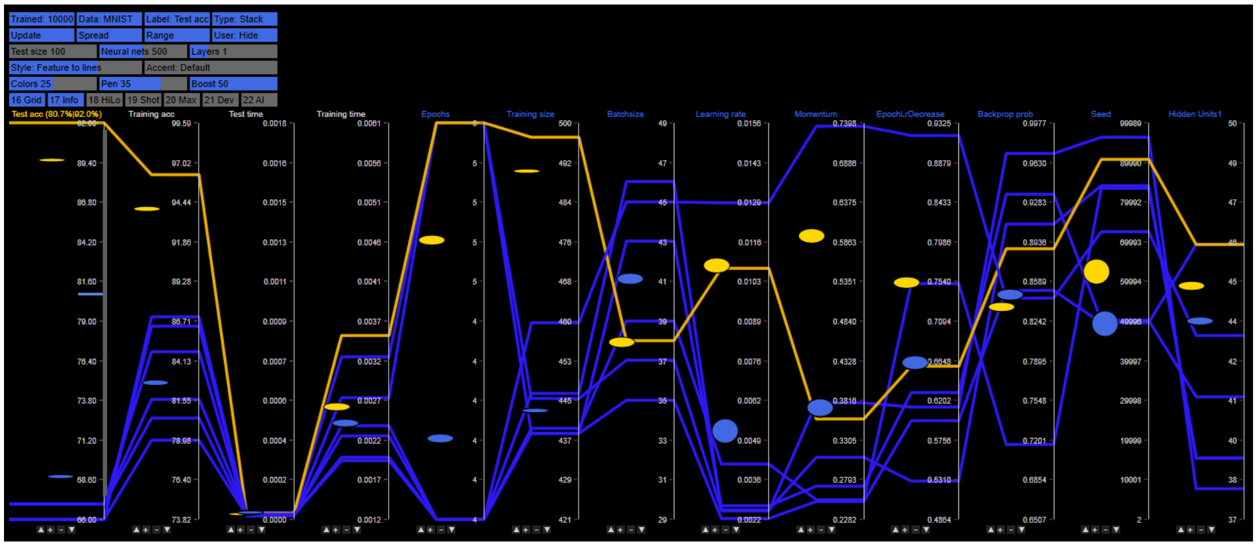
Fixing only the learning rate completely changed the picture also with 10k trained NNs. The test accuracy distribution on the left now has a good range, but is still not normal. Balls far away from each other indicate the most critical features. Also, the colors of some of the blue bars represent networks that are still not learning.

The Normal Distribution



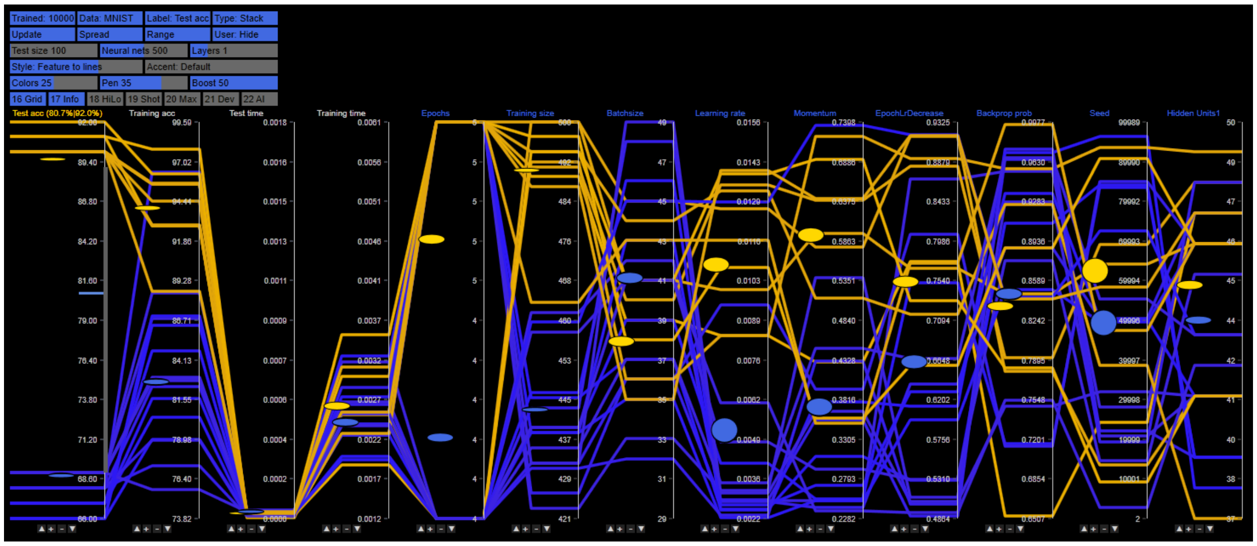
If we fix this as well, we get the normal distribution on the left. This also means we have less blue and gold, now we have some options to get them back. Another striking detail is the training size in every example, the ball near the upper boundary with low variance might tell us that we should increase the training data size, after that more epochs seem good.

Track the Best



Now we get a nice picture by showing only the best and worst NNs as parallel coordinates with features to lines and hiding the rest. The features where all the blue and gold lines separate are the most valuables.

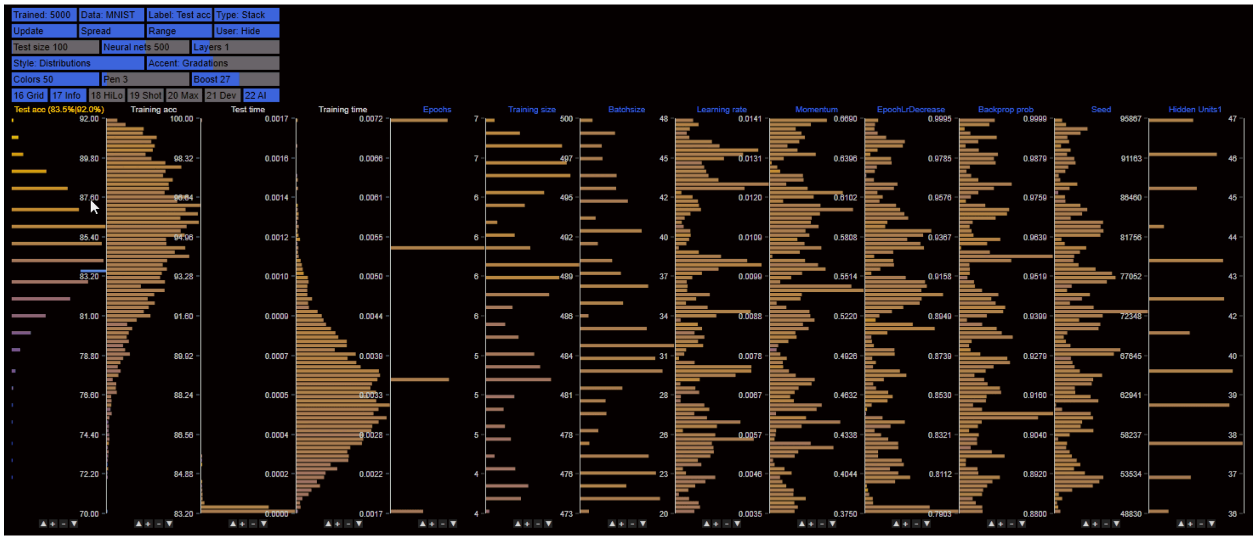
More Confidence for Taking the Next Step



Trained: 10000 Data: MNIST Label: Test acc Type: Stack
Update Spread Range User: Hide
Test size 100 Neural nets 500 Layers 1
Style: Feature to lines Accent: Default
Colors 25 Pen 35 Boost 50
16 Grid 17 Info 18 HiLo 19 Shot 20 Max 21 Dev 22 AI

We can add more lines, but sometimes it becomes a trade-off and we have to take some bad lines or leave some good lines. Or we should track every line for every feature to get a deeper understanding of the relationship between good performing networks.

The Game



Trained: 5000 Data: MNIST Label: Test acc Type: Stack
Update Spread Range User: Hide
Test size 100 Neural nets 500 Layers 1
Style: Distributions Accent: Gradations
Colors 50 Pen 3 Boost 27
16 Grid 17 Info 18 HiLo 19 Shot 20 Max 21 Dev 22 AI

The normal distribution in a real-life scenario can be shifted or squeezed, so we should not expect the perfect picture of a bell-shaped curve. In any case, our goal is to shift the distribution to the top, the perfect distribution for us would be just all at 100%.

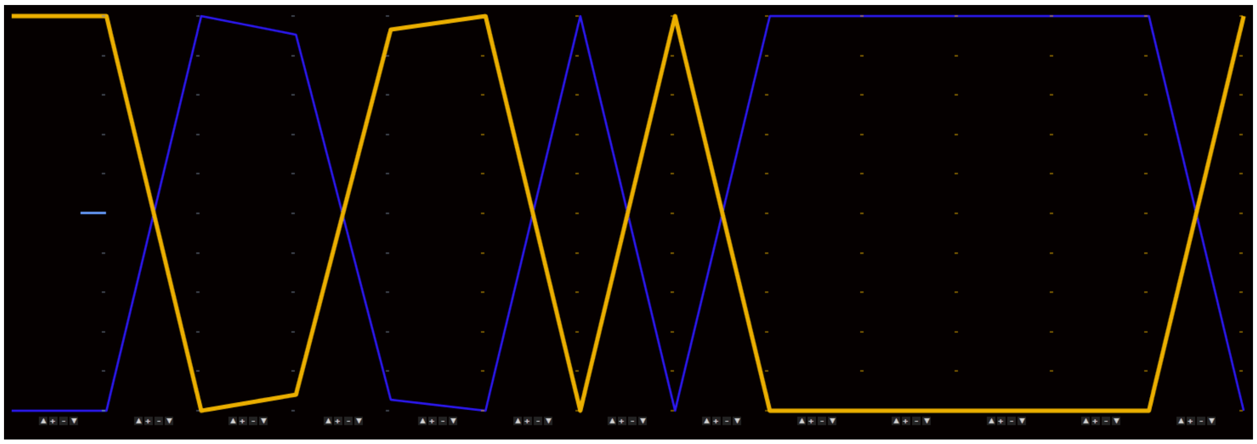
What I show here is more or less a brute force approach to find interesting settings. And it is also a concept to make things work rather than to make them production-ready. However, it is stable and should cover the full range of aspects we will face.

The Evolution of ID24

The next steps could be:
Documentation to SOTA
Profiles
Multi-window support
Web browser support
Server based computing
Multi-player support
New features (users, ranking, more...)
More!

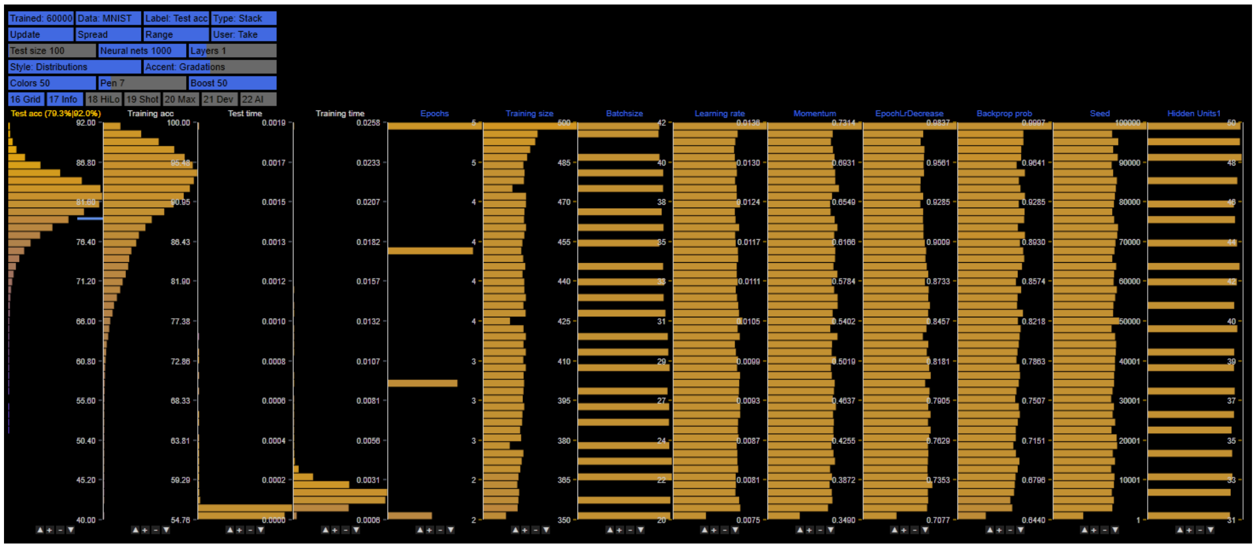
The NOAI Company: goodgame

The Gamer



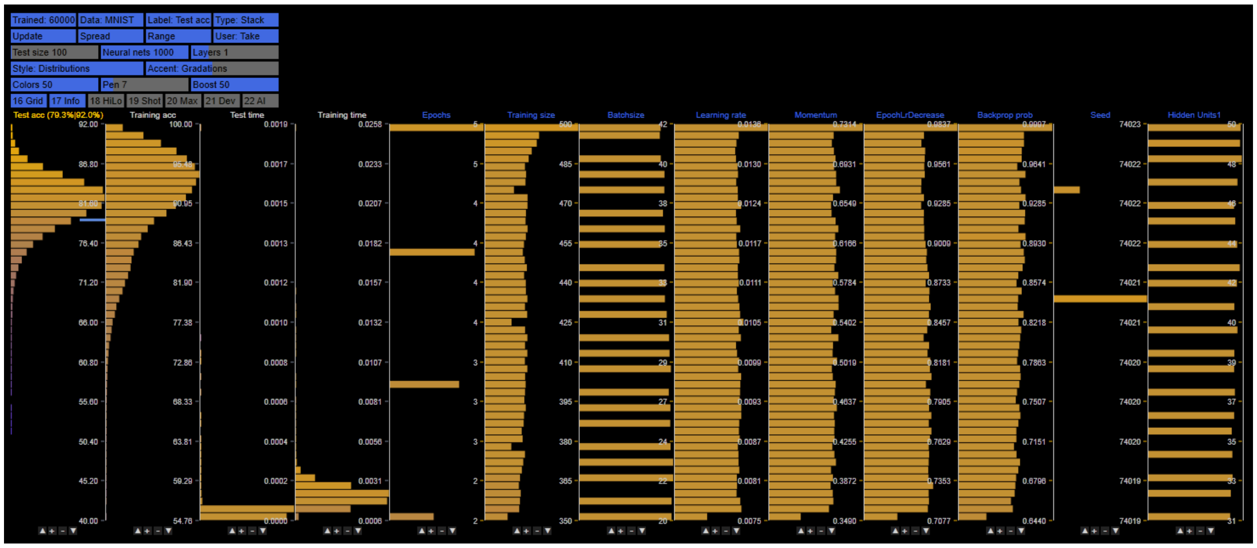
Our workhorse is the gamer, an expert in his game and experienced in solving a variety of different tasks. Without knowing anything about the problem itself, with a maximum reduction of bias and at the same time the highest focus to push the game.

Find Better Seed


Trained: 60000 | Data: MNIST | Label: Test acc | Type: Stack
Update | Spread | Range | User: Take
Test size 100 | Neural nets 1000 | Layers 1
Style: Distributions | Accent: Gradations
Colors 50 | Pen 7 | Boost 50
16 Grid | 17 Info | 18 HiLo | 19 Shot | 20 Max | 21 Dev | 22 AI

The player will find things like the better seed, and I am still not sure, but I cannot say the opposite. Higher numbers seem to perform slightly better in a lot of cases. Which makes sense in a weird way.

Good Seed 74021?


Trained: 60000 | Data: MNIST | Label: Test acc | Type: Stack
Update | Spread | Range | User: Take
Test size 100 | Neural nets 1000 | Layers 1
Style: Distributions | Accent: Gradations
Colors 50 | Pen 7 | Boost 50
16 Grid | 17 Info | 18 HiLo | 19 Shot | 20 Max | 21 Dev | 22 AI

74021 was a very good seed, and although I cannot explain it exactly, higher seeds produce better results than lower seeds. I saw this very often and it was unexpected.

Suppose the competitor gets the same result as we do, but doesn't know about the better seed stuff, then we could do the better job.

It also seems possible to prepare the dataset to remove bad examples. There are a lot of things we can do, but probably the best thing would be to combine it with other ideas.

If this is a bet, I bet on humans. And they are better with AI. The extra “NOAI” didn't mean no AI. It comes from the term NoSQL, which stands for Not Only SQL.
There is a lot more to talk about, however the 25 MB limit of gmail stops more cool animations. The id24v3.cs code as WPF demo is attached. May it takes some seconds to download the dataset. The path can be changed on top of the code.

But the question I have to you James:
Should we start this company and move the world forward by making machines work like no one has ever seen before? This tool can also prove density.

Very hot:
https://twitter.com/jaschasd/status/1756930242965606582

TK