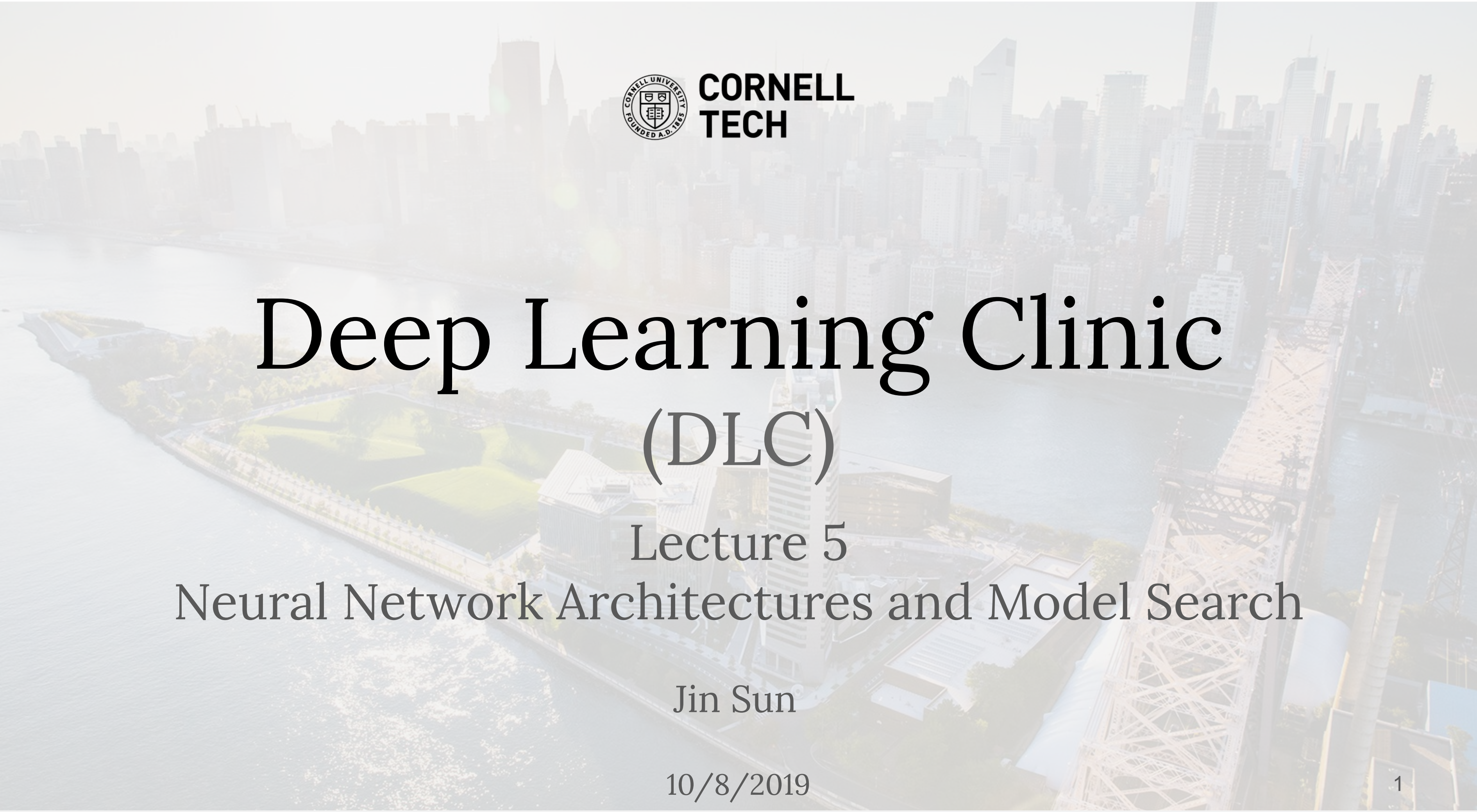


CORNELL TECH

Deep Learning Clinic

(DLC)

Lecture 5
Neural Network Architectures and Model Search

Jin Sun

10/8/2019

Today - Neural Network Architectures and Model Search

- **Neural Network Architecture Design**
- Hyperparameters and Network Sensitivity
- Optimize Hyperparameters Systematically
- Automatic Model Search - AutoML

Based on course materials from http://cs231n.github.io/

Design a NN model for your problem

Things to consider:

- ❏ **The characteristic of your data**
- ❏ Data input/output space
- ❏ Budget on hardwares, training time...

Recap: Data and Neural Network Models

Static Data

Convolutional Neural Networks

Dynamic Data

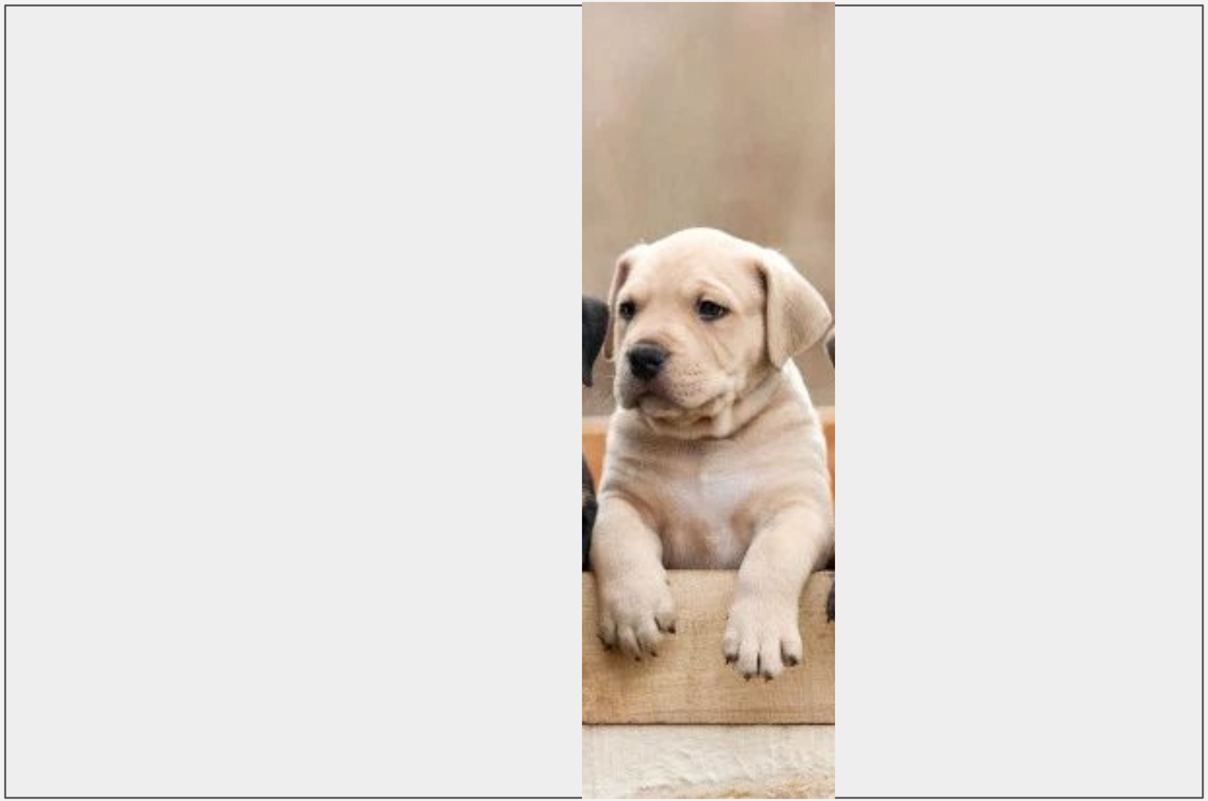
Recurrent Neural Networks

Unsupervised Data

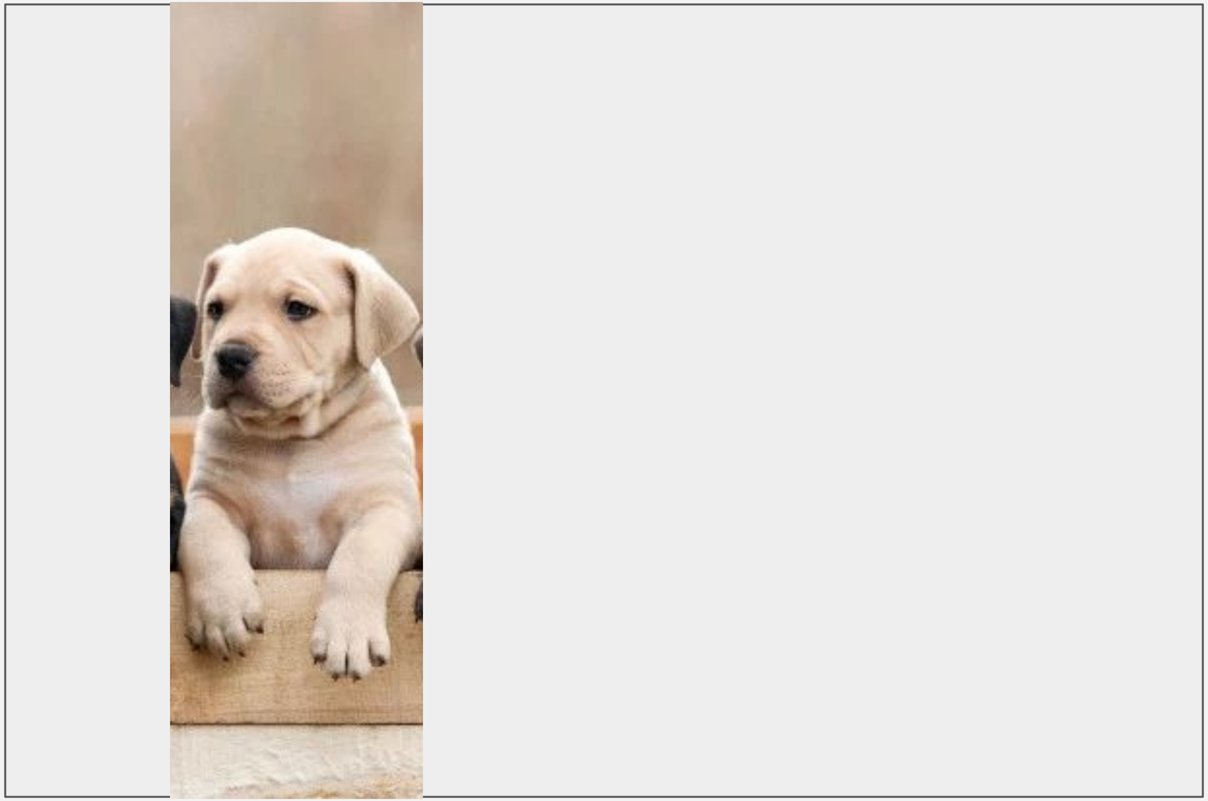
Generative Neural Networks

Static Data - Image

Static Data - Translation invariance in images

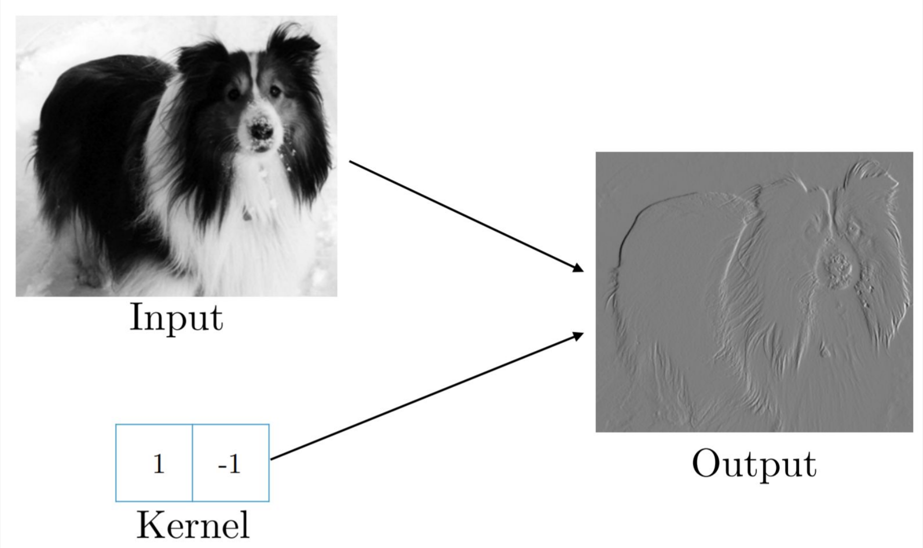


Static Data - Translation invariance in images

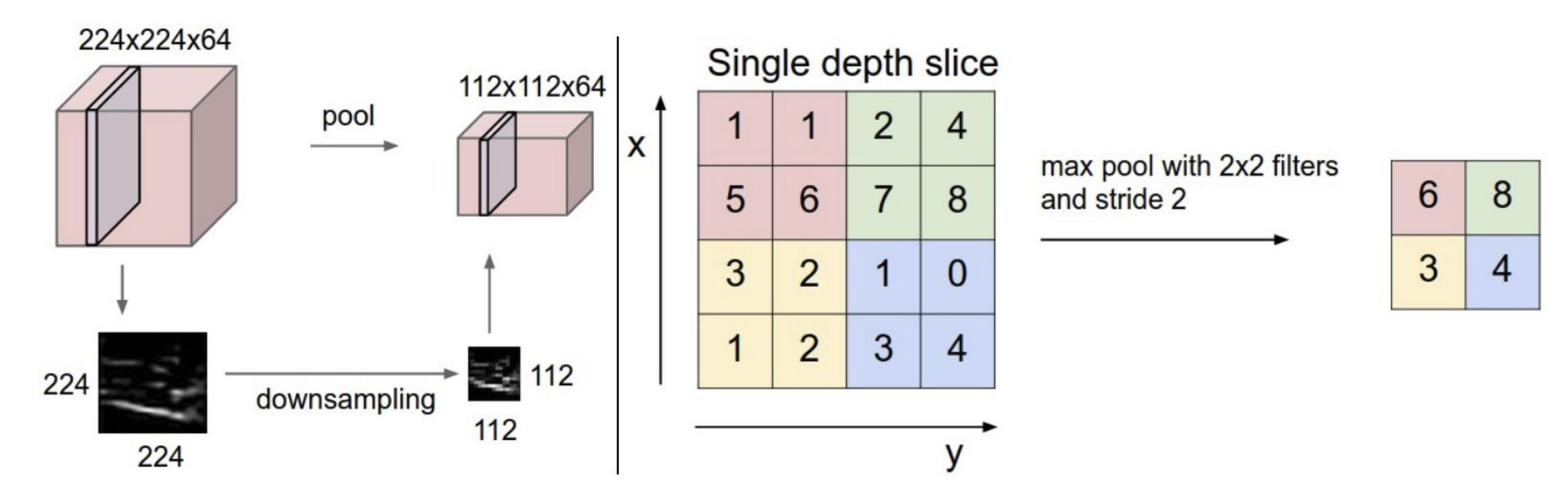


Convolution

A local operation that extracts information from data.



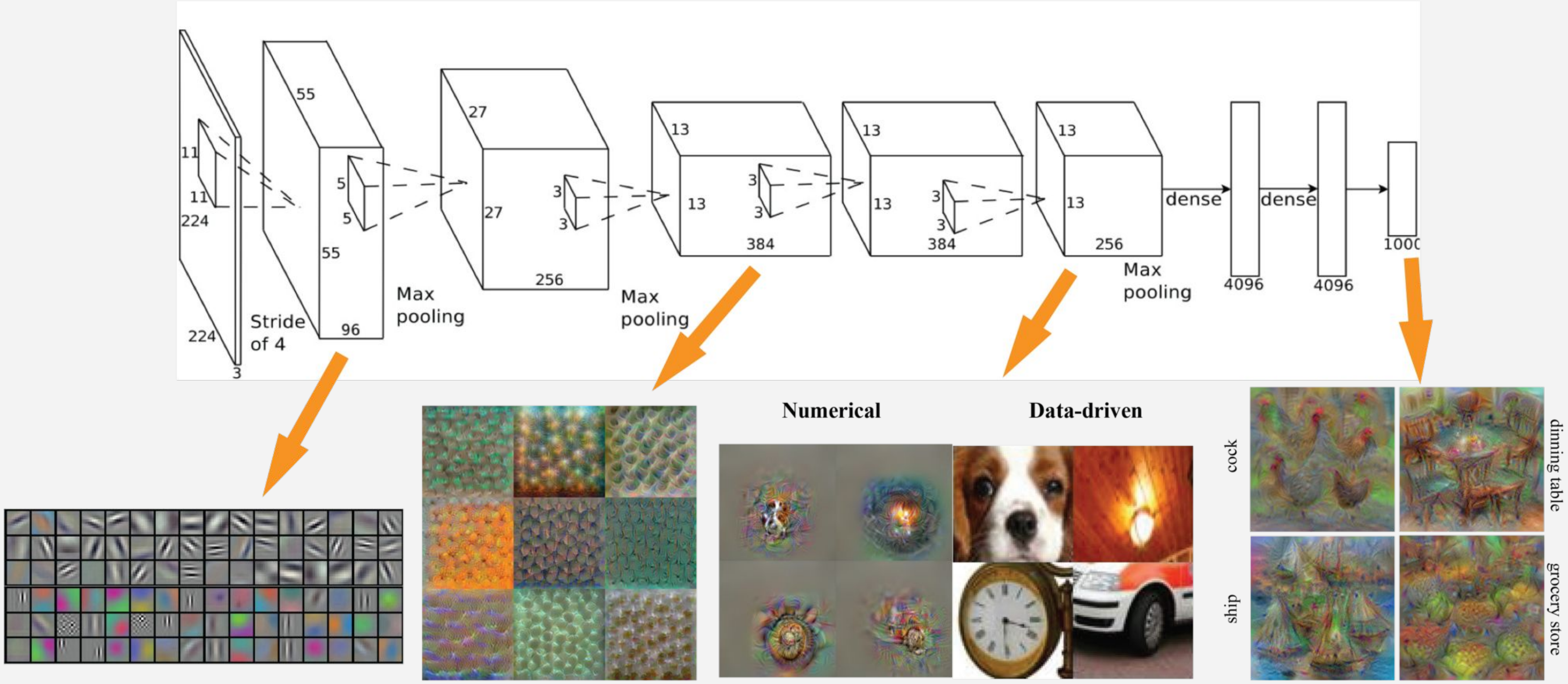
Pooling



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

Convolutional Networks



Conv 1: Edge+Blob

Conv 3: Texture

Conv 5: Object Parts

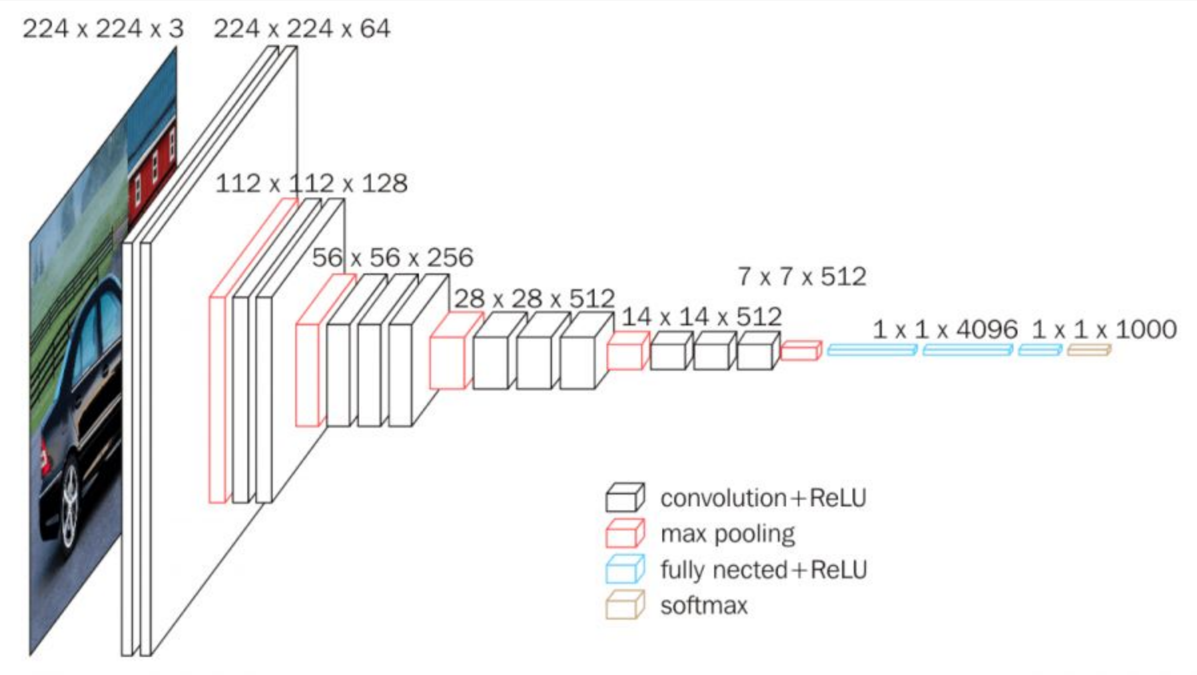
Fc8: Object Classes

Design a NN model for your problem

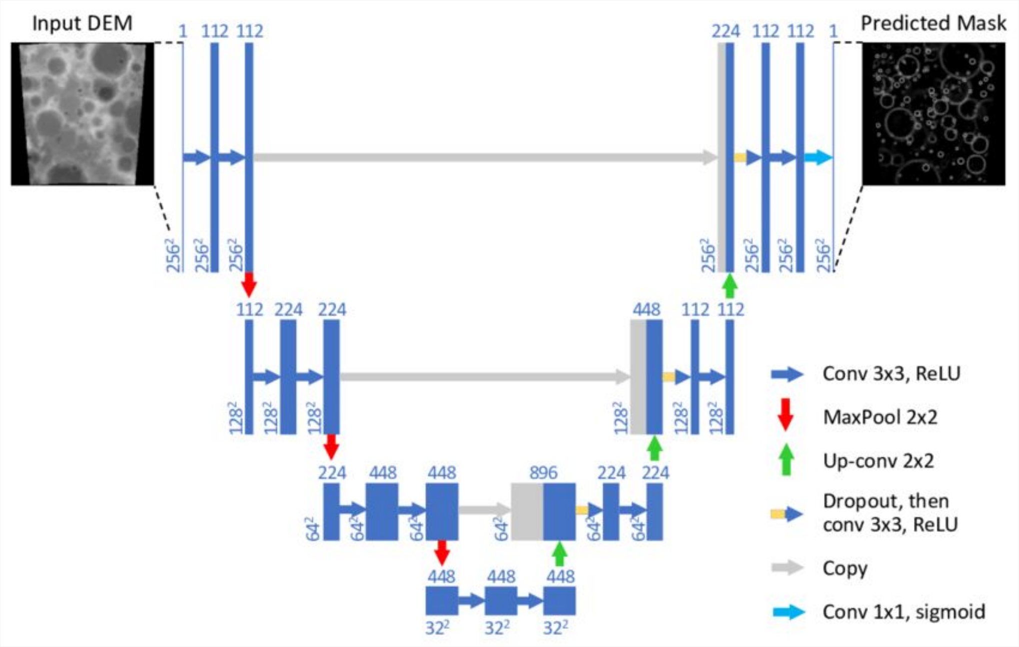
Things to consider:

- ❏ The characteristic of your data
- ❏ **Data input/output space**
- ❏ Budget on hardwares, training time…

Predicting a category

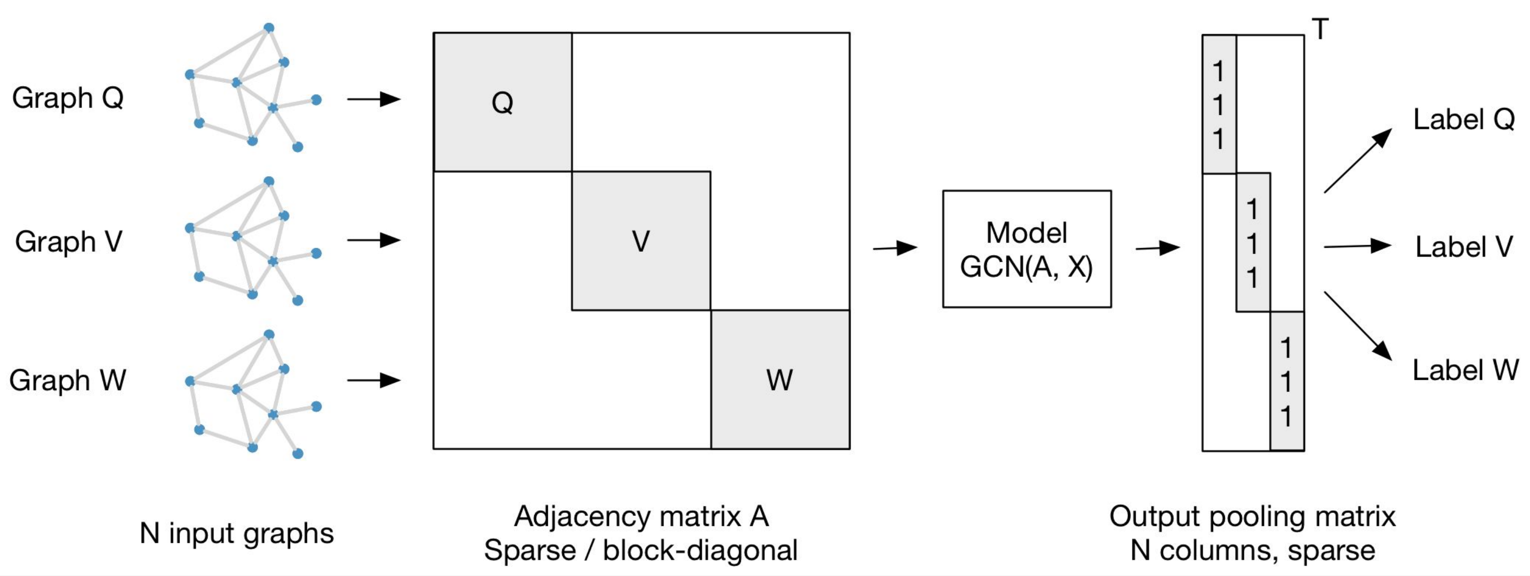


Dense predictions



U-Net

Input is a graph



Graph Conv Net

https://github.com/tkipf/gcn

Design a NN model for your problem

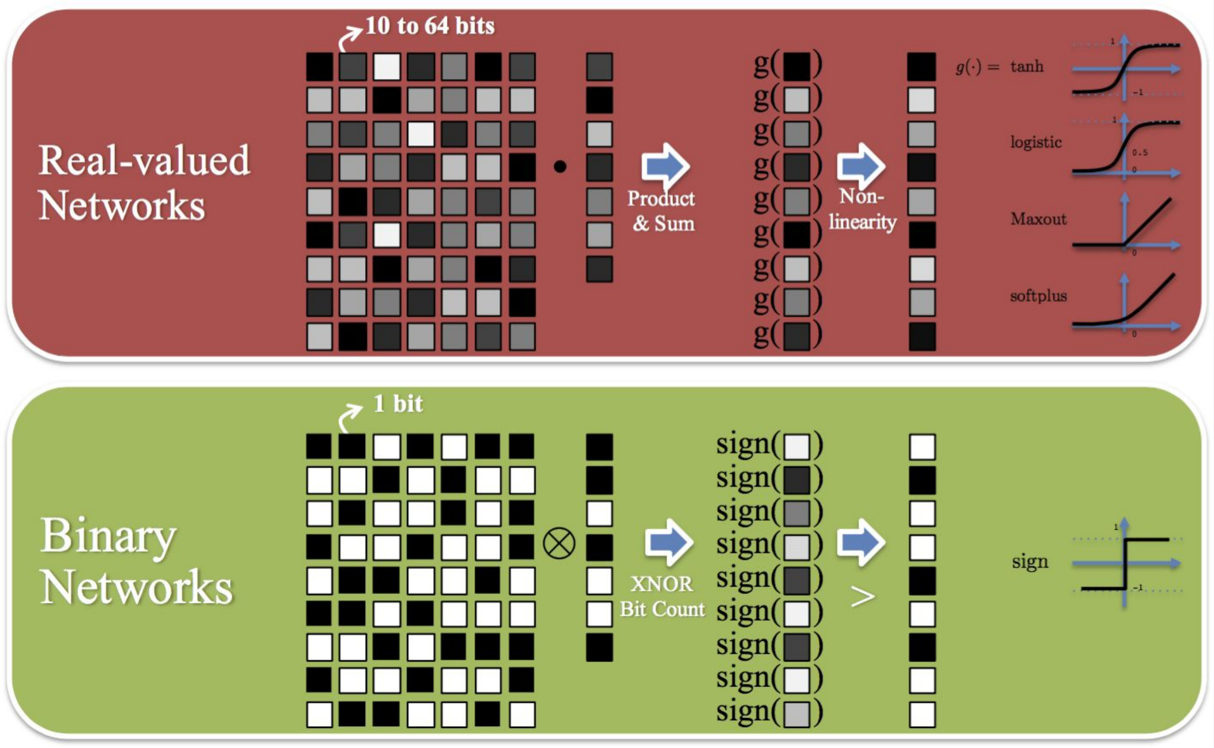
Things to consider:

❑ The characteristic of your data

❑ Data input/output space

❑ **Budget on hardwares, training time…**

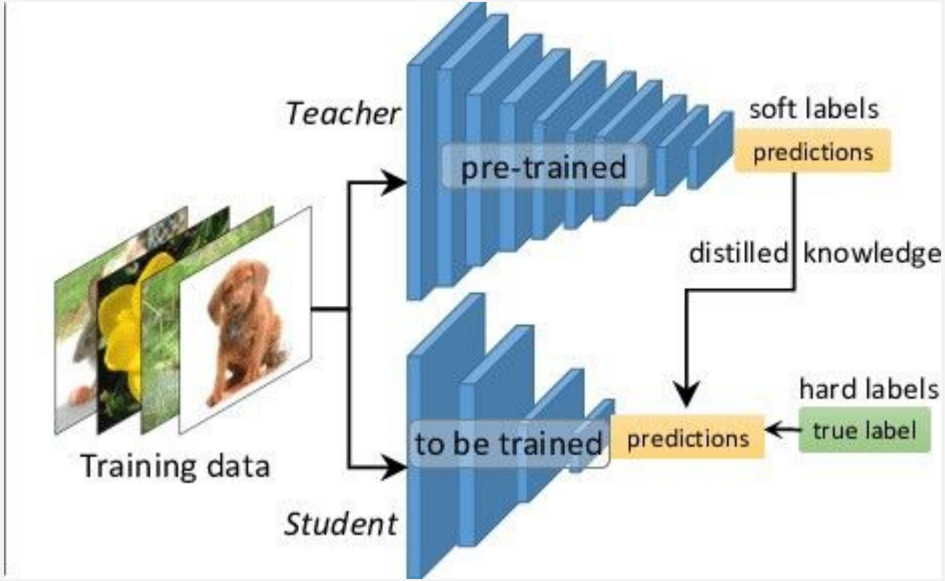
Binary Networks



https://mohitjain.me/2018/07/14/bnn/

Knowledge Distillation

Train a smaller network (Student) by learning from a more powerful network (Teacher).



https://towardsdatascience.com/knowledge-distillation-simplified-dd4973dbc764

Today - Neural Network Architectures and Model Search

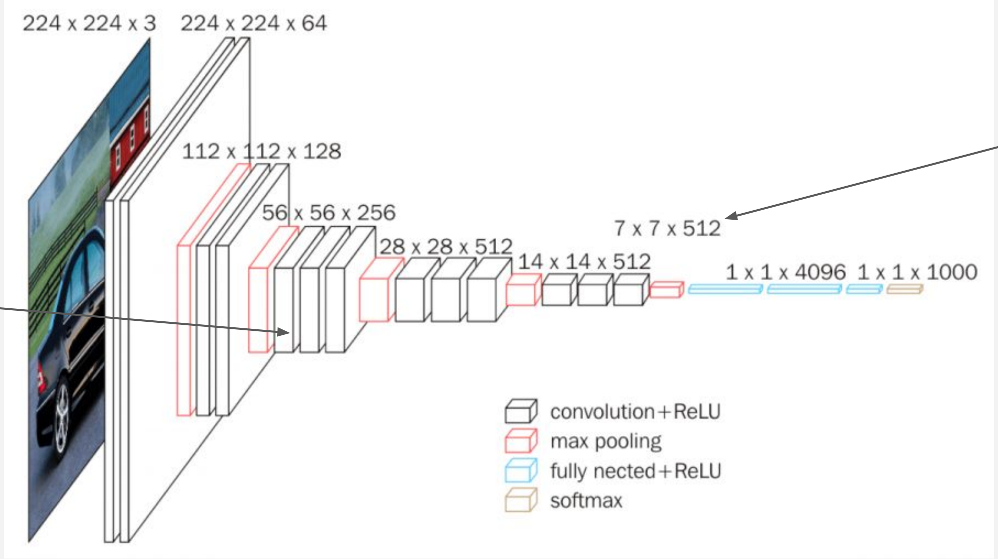
- Neural Network Architecture Design
- **Hyperparameters and Network Sensitivity**
- Optimize Hyperparameters Systematically
- Automatic Model Search - AutoML

A recommended read:

Bengio, Yoshua. "Practical recommendations for gradient-based training of deep architectures." *Neural networks: Tricks of the trade*. Springer, Berlin, Heidelberg, 2012. 437-478.

Neural Network Model Search

Even with a targeted network architecture, there are still a lot of flexibility in the exact configuration of your neural network framework.



How many filters?

How many layers?

Hyperparameters

Learning rate: controls how much you update the model's weights.

Batch size: how many data you put into the model each time to calculate the gradient.

Regularization: e.g., weight decay, L1.

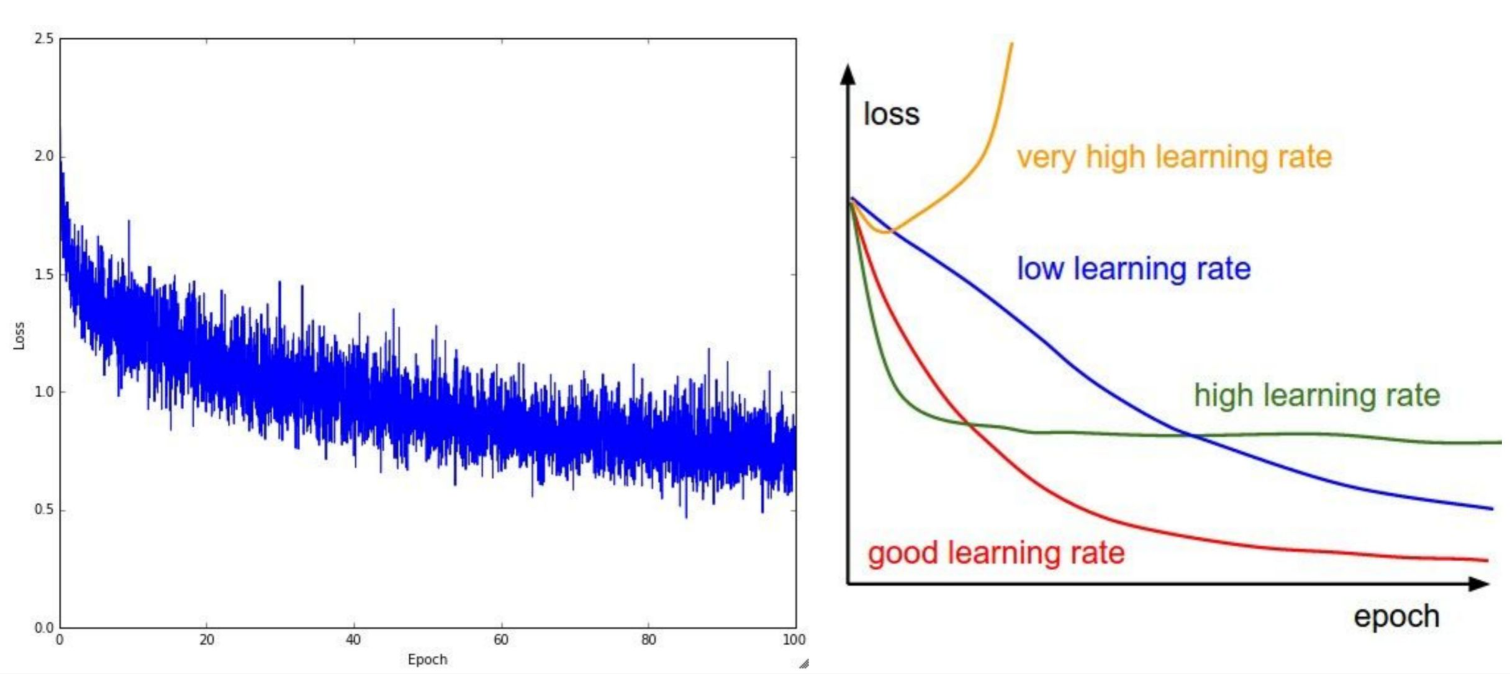
Network structure: number of layers, activation functions, etc.

...

Even **Data**: it is almost always guaranteed that a model would get better with more data.

Sensitivity of Networks to Hyperparameters

Learning Rate (we will cover a lot on this in Lecture - Tricks)



Sensitivity of Networks to Hyperparameters

Batchsize

$$\theta = \theta - \alpha \bigtriangledown_\theta E[J(\theta)]$$

Decrease learning rate is to reduce the noise of each batch’s gradient update.

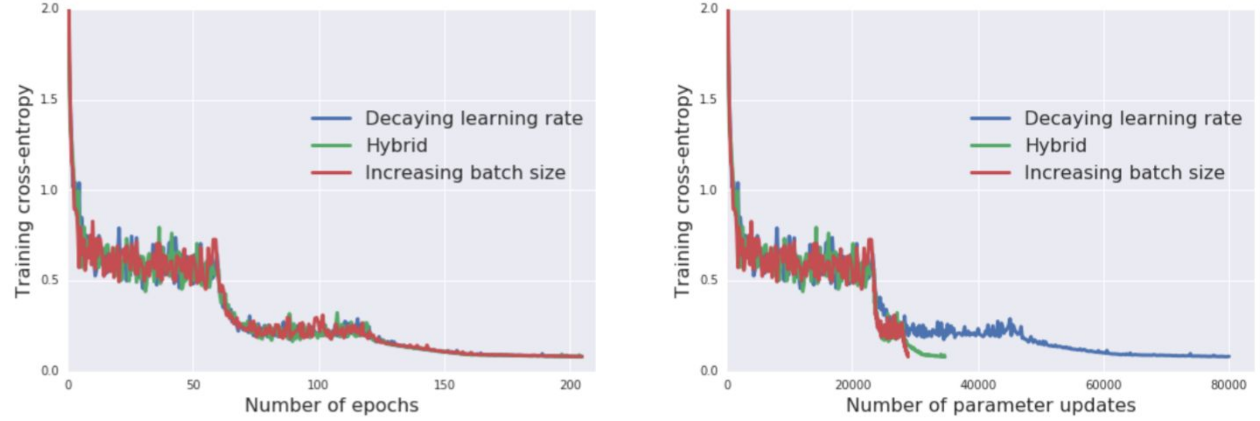
=

Increase batchsize to reduce the noise in each batch.

Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." *arXiv preprint arXiv:1711.00489*(2017).

Sensitivity of Networks to Hyperparameters

Batchsize



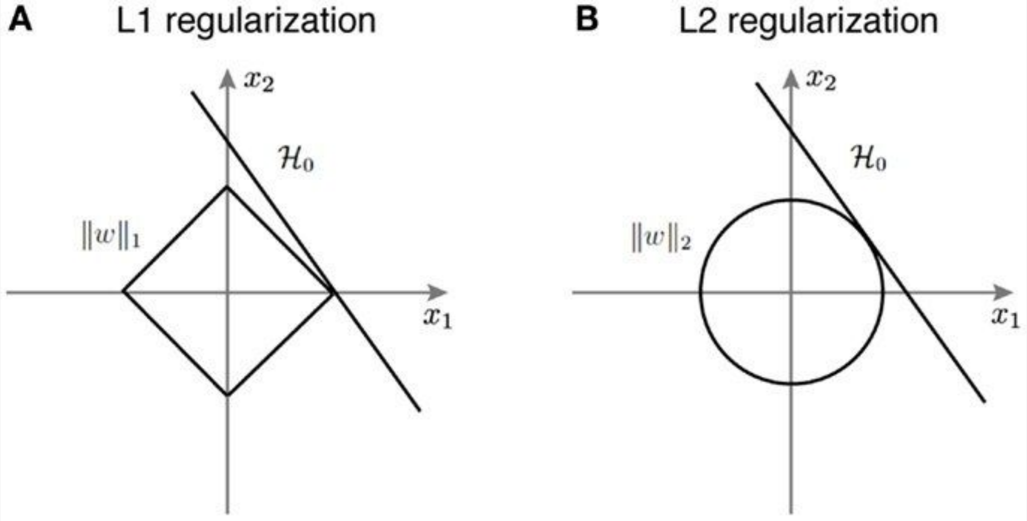
(a) (b)

Figure 2: Wide ResNet on CIFAR10. Training set cross-entropy, evaluated as a function of the number of training epochs (a), or the number of parameter updates (b). The three learning curves are identical, but increasing the batch size reduces the number of parameter updates required.

Smith, Samuel L., et al. "Don't decay the learning rate, increase the batch size." *arXiv preprint arXiv:1711.00489*(2017).

Sensitivity of Networks to Hyperparameters

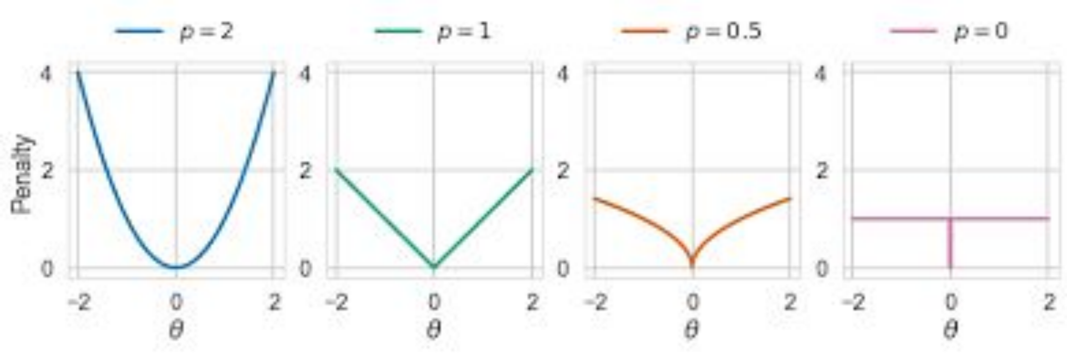
Regularization



https://www.quora.com/Why-is-L1-regularization-supposed-to-lead-to-sparsity-than-L2

Sensitivity of Networks to Hyperparameters

Regularization

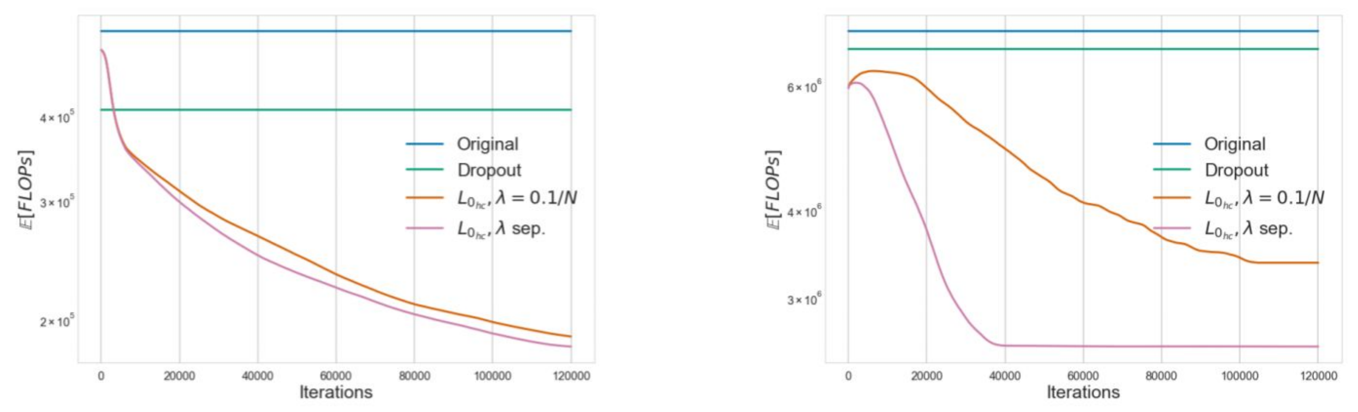


Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).

Sensitivity of Networks to Hyperparameters

Regularization

Training with L0 regularization = model pruning



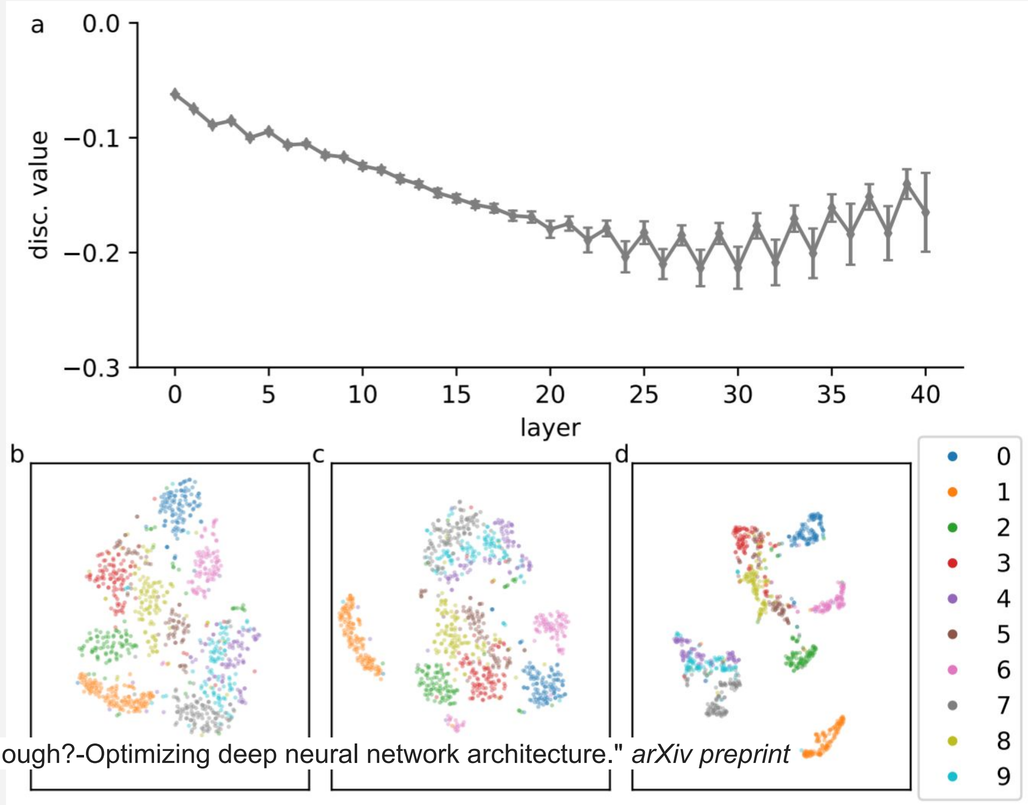
(a) Expected FLOPs at the MLP.

(b) Expected FLOPs at LeNet5.

Louizos, Christos, Max Welling, and Diederik P. Kingma. "Learning Sparse Neural Networks through $ L_0 $ Regularization." *arXiv preprint arXiv:1712.01312* (2017).

Sensitivity of Networks to Hyperparameters

Number of Layers/Depth



Schilling, Achim, et al. "How deep is deep enough?-Optimizing deep neural network architecture." *arXiv preprint arXiv:1811.01753* (2018).

Sensitivity of Networks to Hyperparameters

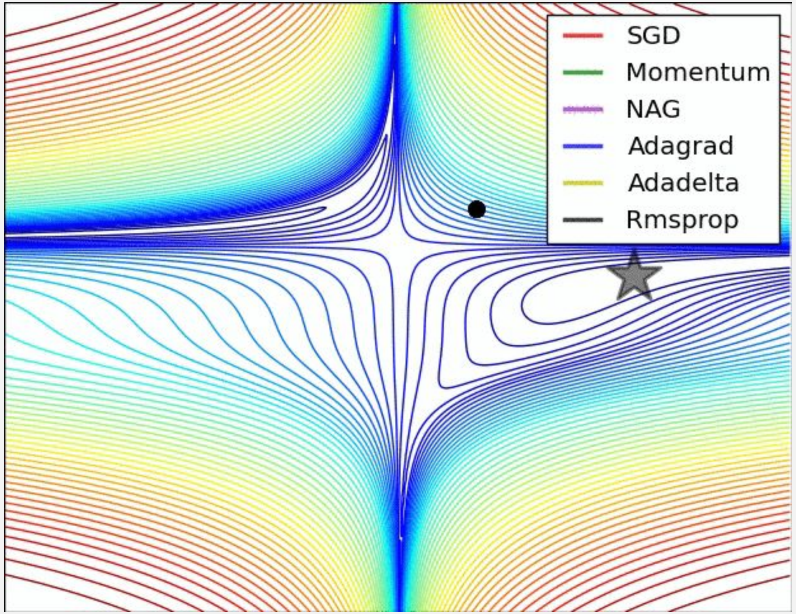
Number of Layers/Depth

Deeper networks are harder to train.

Consider add normalization (BatchNorm and its variants) and ResNet modules to the network for **stability** and **strong gradients**.

Sensitivity of Networks to Hyperparameters

Optimizer



http://ruder.io/optimizing-gradient-descent/

Sensitivity of Networks to Hyperparameters

Optimizer

```
torch.optim.Adadelta(params, lr=1.0, rho=0.9, eps=1e-06, weight_decay=0)

torch.optim.Adagrad(params, lr=0.01, lr_decay=0, weight_decay=0, initial_accumulator_value=0)

torch.optim.Adam(params, lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False)

torch.optim.ASGD(params, lr=0.01, lambd=0.0001, alpha=0.75, t0=1000000.0, weight_decay=0)

torch.optim.SGD(params, lr=<required parameter>, momentum=0, dampening=0, weight_decay=0, nesterov=False)

torch.optim.LBFGS(params, lr=1, max_iter=20, max_eval=None, tolerance_grad=1e-05, tolerance_change=1e-09,
history_size=100, line_search_fn=None)

...
```

Today - Neural Network Architectures and Model Search

- Neural Network Architecture Design
- Hyperparameters and Network Sensitivity
- **Optimize Hyperparameters Systematically**
- Automatic Model Search - AutoML

Before Your Start Hyperparameter Search

Sanity Check!

Use a **small portion** of the data, remove all the fancy stuff.

Train your model, you should be able to **overfit** the model to your small data: very high accuracy on your training data (not validation data).

If not, there is some bug in your pipeline.

This is Probably Your Current Approach

Use a set of default hyperparameter settings.

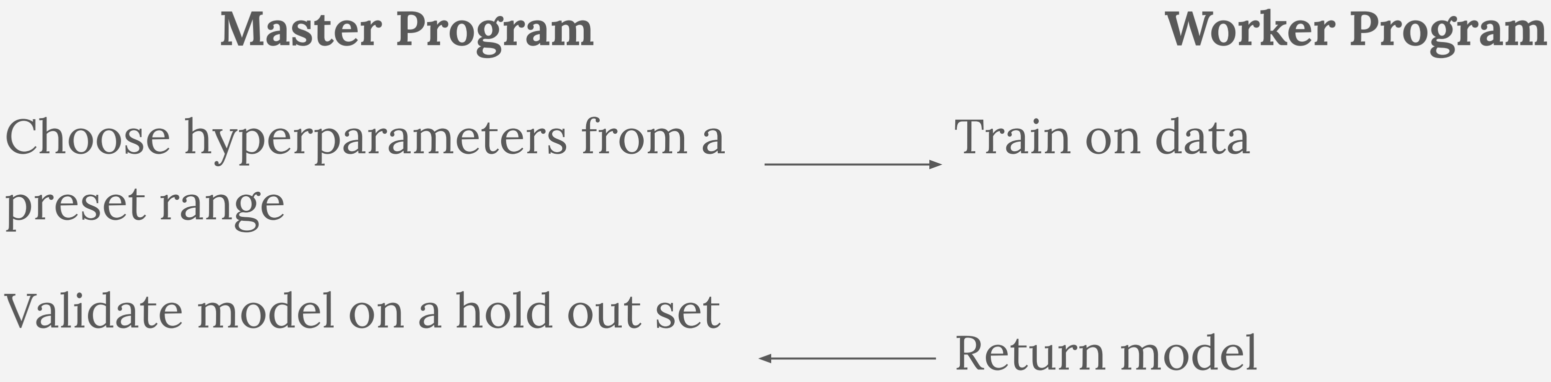
Train a model.

Check for metric.

Randomly change things.

Repeat above until satisfaction.

Optimize Hyperparameters Systematically



Master Program

Worker Program

Choose hyperparameters from a preset range

Train on data

Validate model on a hold out set

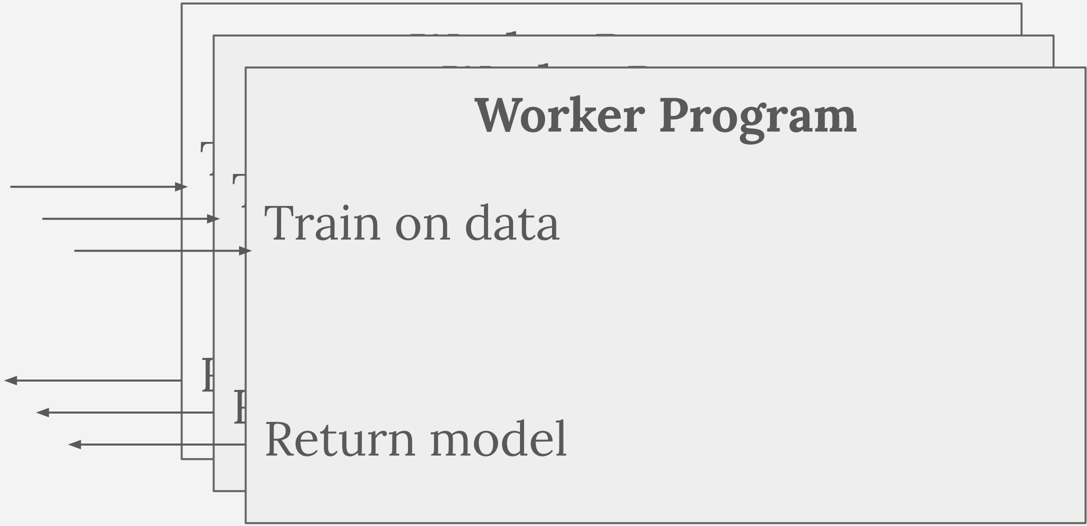
Return model

Optimize Hyperparameters Systematically

Master Program

Choose hyperparameters from a preset range

Validate model on a hold out set

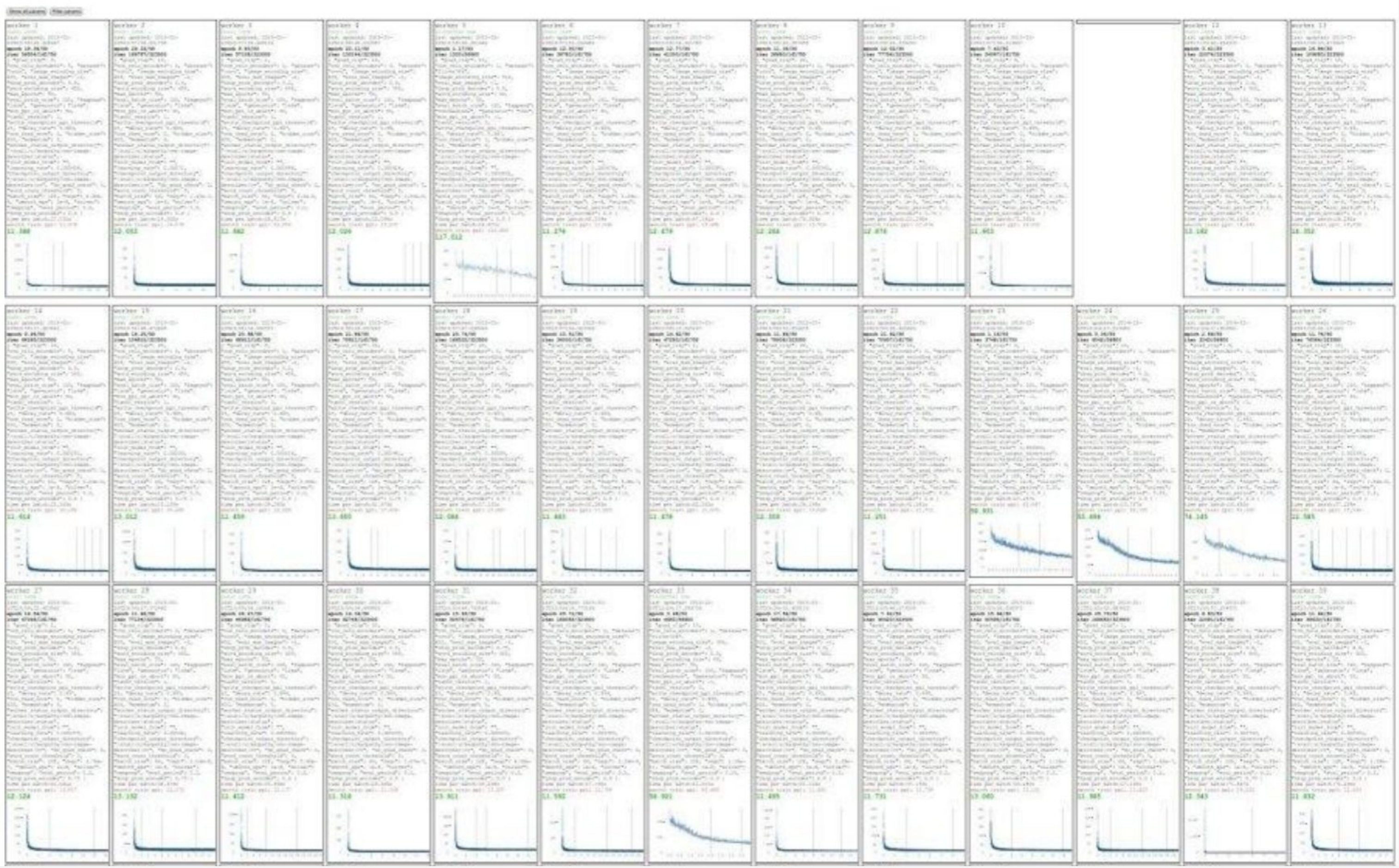


Worker Program

Train on data

Return model

Scalable

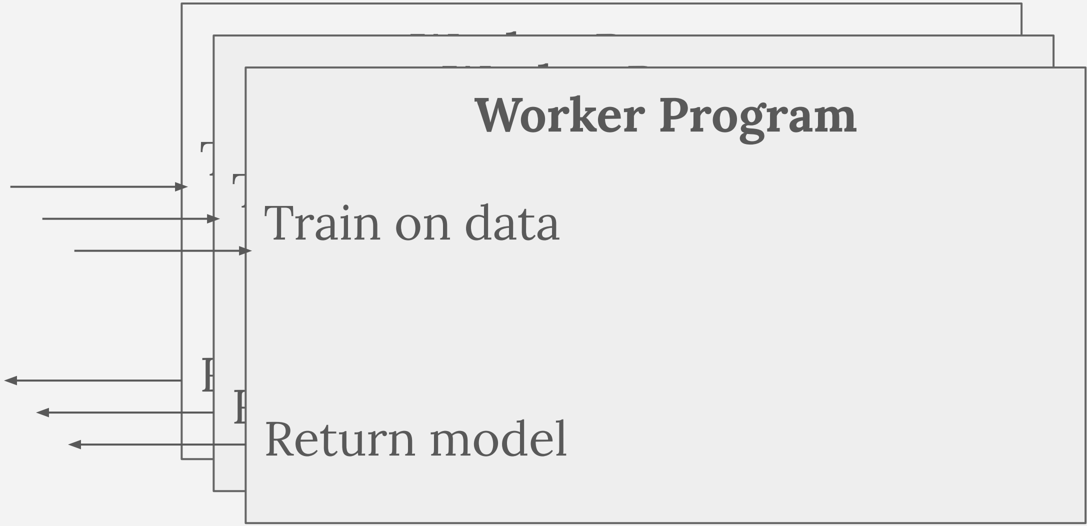


Optimize Hyperparameters Systematically

Master Program

Choose hyperparameters from a preset range

Validate model on a hold out set



Worker Program

Train on data

Return model

Optimize Hyperparameters Systematically

Choose your **train and validation set**.

Many dataset comes with default ones, but sometimes it is good to pick a **subset for fast prototyping**.

7:3 train to validation ratio is traditionally suggested.

8:2 is also more popular now.

N-Fold **Cross Validation?**

In early stage of hyperparameter tuning it is enough to use a single set.

Optimize Hyperparameters Systematically

Set a **range** of your parameters:

Learning rate: [1e-3, 1e-6] with Adam, might be different for diff optimizer.

Batchsize: [16, 256], depending on the memory size.

Regularizer: {(L0), L1, L2}

...

Almost always good to work in **log space** for continuous values.

Optimize Hyperparameters Systematically

Almost always good to work in **log space** for continuous values

Learning rate values to be picked:

1e-3, 1e-4, 1e-5, 1e-6, ...

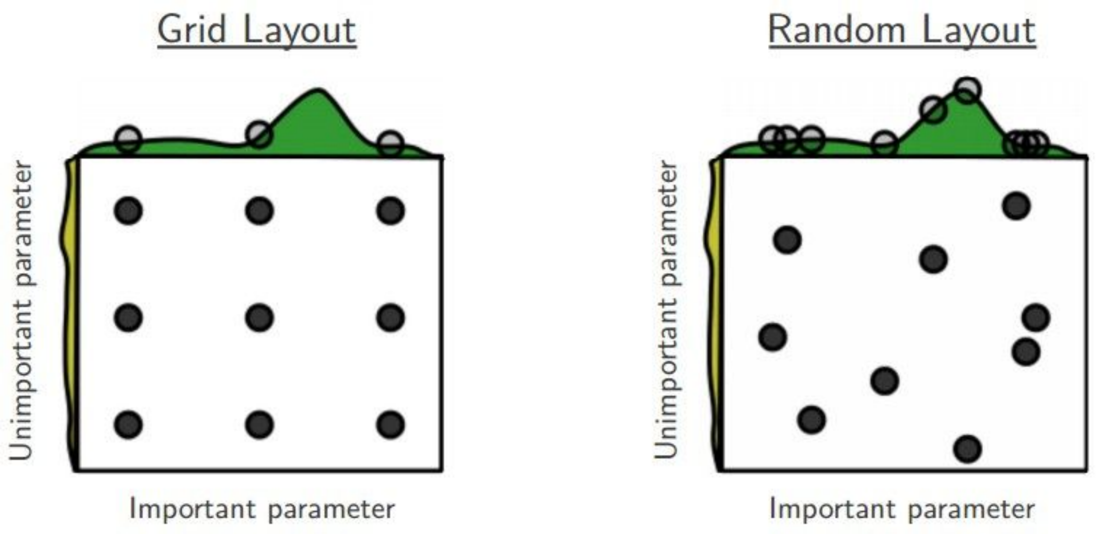
Batchsize:

8, 16, 32, 64, ...

Optimize Hyperparameters Systematically

Grid search vs random search in the parameter range

E.g., [1e-3, 1e-6]

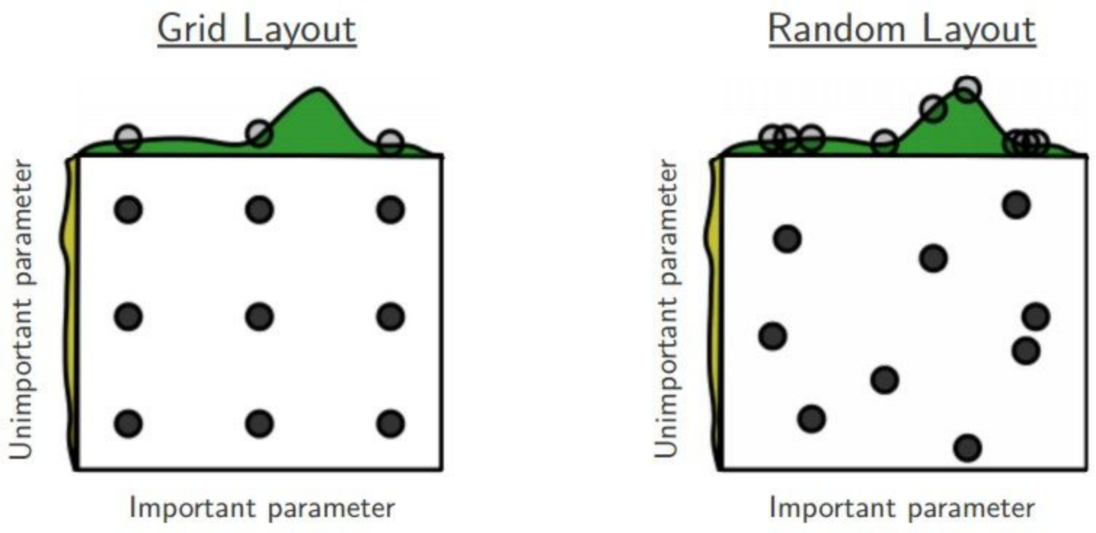


http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Optimize Hyperparameters Systematically

Grid search vs random search in the parameter range

Because the parameters are not equally good in the parameter space



http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf

Optimize Hyperparameters Systematically

Coarse to Fine Process

Start with few epochs, on a wider range of parameters

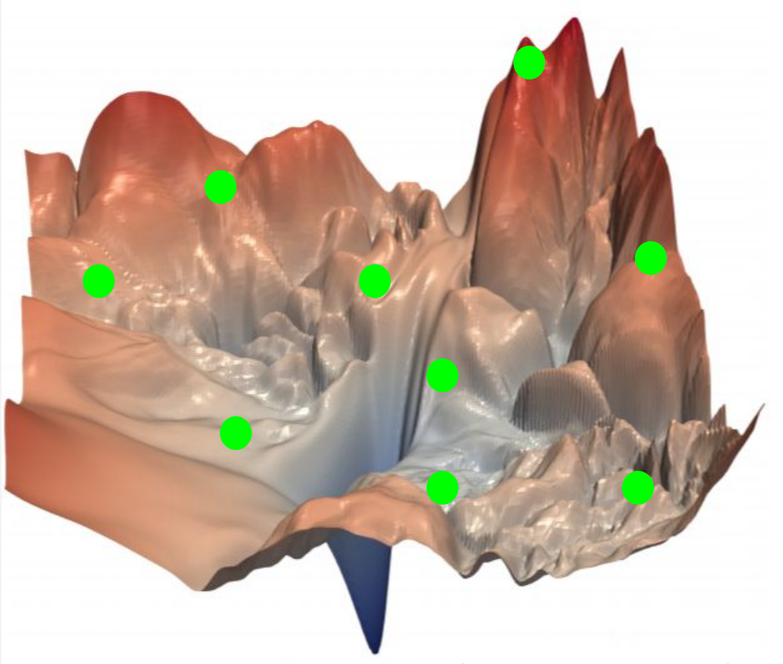
Then use finer scale and run more epochs

Optimize Hyperparameters Systematically

Coarse to Fine Process

Always keep in mind this is a

non-convex optimization



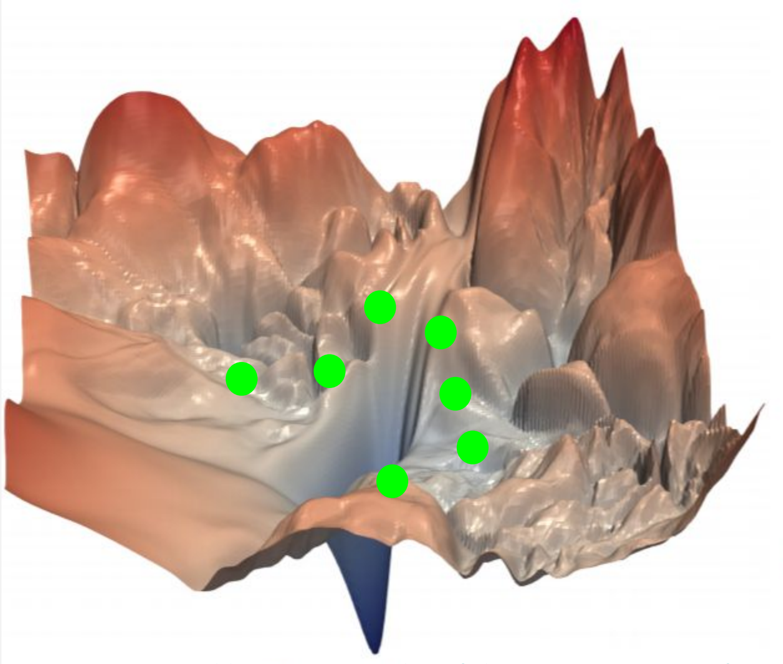
https://arxiv.org/pdf/1712.09913.pdf

Optimize Hyperparameters Systematically

Coarse to Fine Process

Always keep in mind this is a

non-convex optimization



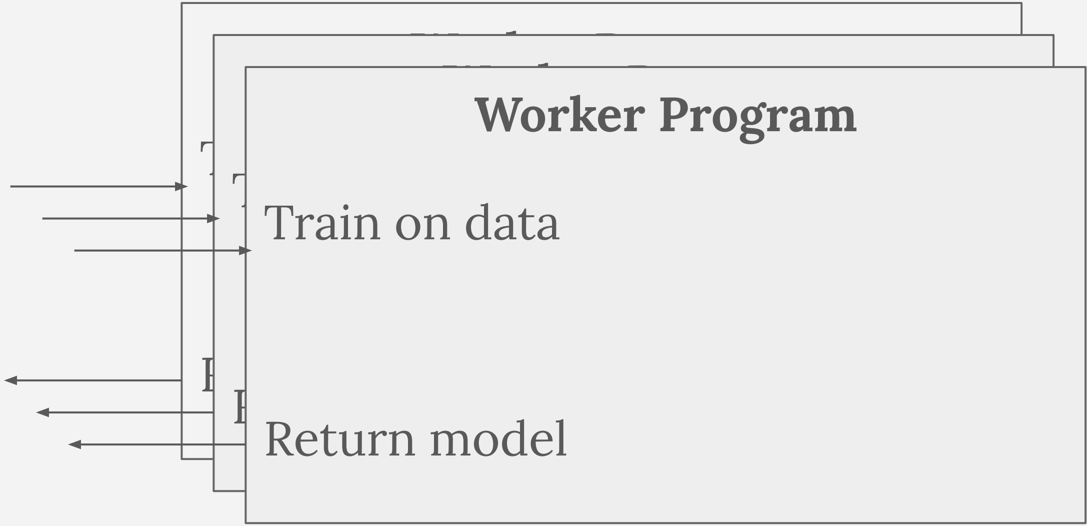
https://arxiv.org/pdf/1712.09913.pdf

Optimize Hyperparameters Systematically

Master Program

Choose hyperparameters from a preset range

Validate model on a hold out set



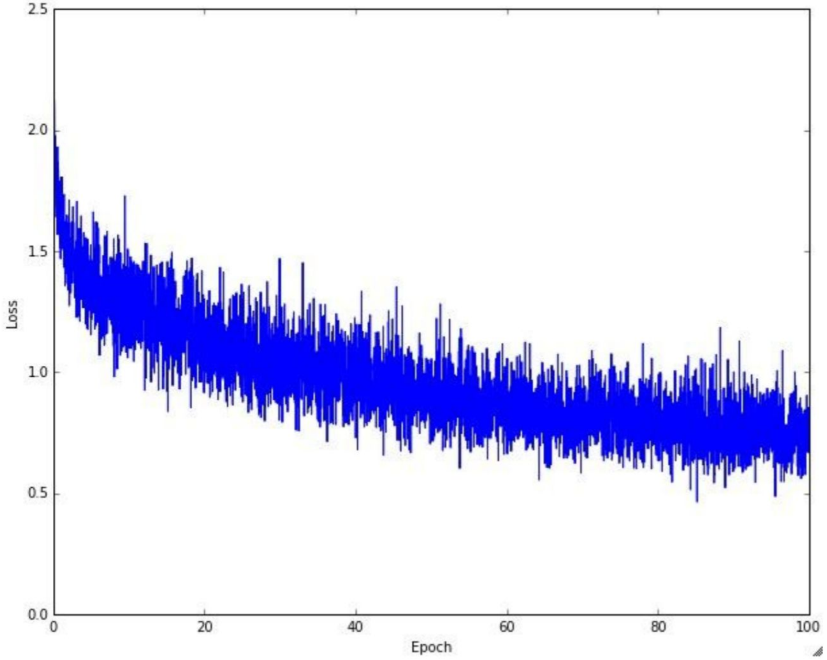
Worker Program

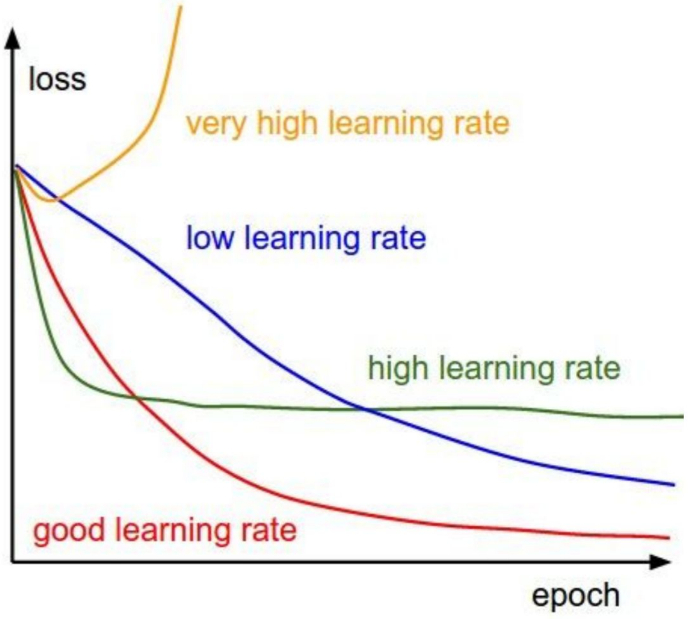
Train on data

Return model

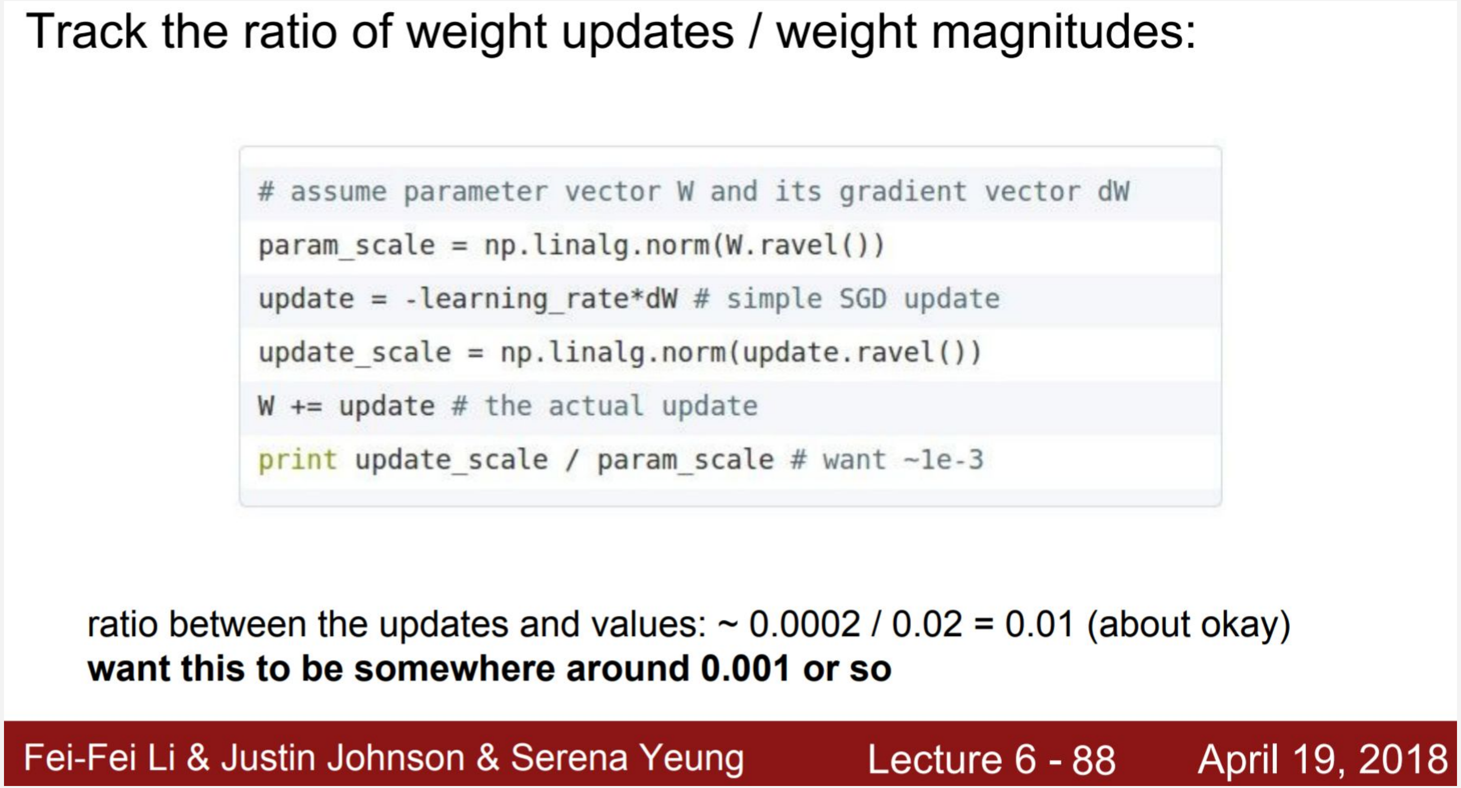
What to check during the training process?

[Major] Loss values comparison between train and validation





Look deeper



Track the ratio of weight updates / weight magnitudes:

```
# assume parameter vector W and its gradient vector dW
param_scale = np.linalg.norm(W.ravel())
update = -learning_rate*dW # simple SGD update
update_scale = np.linalg.norm(update.ravel())
W += update # the actual update
print update_scale / param_scale # want ~1e-3
```

ratio between the updates and values: ~ 0.0002 / 0.02 = 0.01 (about okay)
want this to be somewhere around 0.001 or so

Fei-Fei Li & Justin Johnson & Serena Yeung Lecture 6 - 88 April 19, 2018

What to check during the training process?

Visual inspection of predicted results might be also useful.

For example, take a look at generated images, depth maps, semantic segmentation maps.

There might be 2.3% of corrupted data in your dataset! You cannot tell this by just looking at metric numbers.

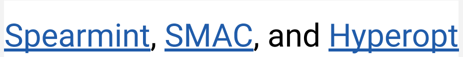
Optimize Hyperparameters Systematically

The hard way:

Bayesian Hyperparameter Optimization

The Bayesian view:

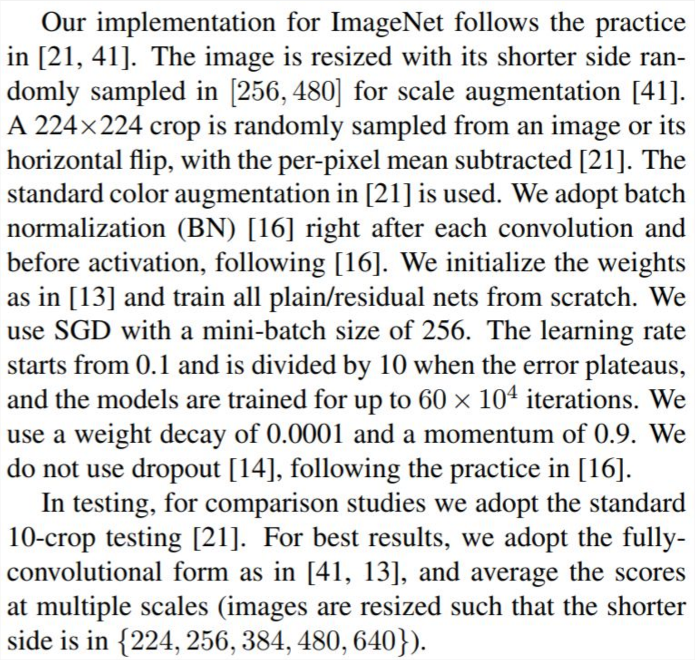
Each hyperparameter is a random variable from some distribution. The goal is to find the optimal set of hyperparameters from sampling from the distribution and observe their performance.

Example methods: 

Spearmint, SMAC, and Hyperopt

Remember to consult the literature

Try to take a look at state-of-the-art related work.



Our implementation for ImageNet follows the practice in [21, 41]. The image is resized with its shorter side randomly sampled in $[256, 480]$ for scale augmentation [41]. A 224×224 crop is randomly sampled from an image or its horizontal flip, with the per-pixel mean subtracted [21]. The standard color augmentation in [21] is used. We adopt batch normalization (BN) [16] right after each convolution and before activation, following [16]. We initialize the weights as in [13] and train all plain/residual nets from scratch. We use SGD with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error plateaus, and the models are trained for up to 60×10^4 iterations. We use a weight decay of 0.0001 and a momentum of 0.9. We do not use dropout [14], following the practice in [16].

In testing, for comparison studies we adopt the standard 10-crop testing [21]. For best results, we adopt the fully-convolutional form as in [41, 13], and average the scores at multiple scales (images are resized such that the shorter side is in $\{224, 256, 384, 480, 640\}$).

He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Get the Most From Your Models

Model Ensembles

Instead of having a single trained model, you could generate a number of similar models and get a slightly better overall performance.

- Same model with different initializations.

Train model with the same set of hyperparameters, but with different random initialized weights.

http://cs231n.github.io/neural-networks-3/

Get the Most From Your Models

Model Ensembles

Instead of having a single trained model, you could generate a number of similar models and get a slightly better overall performance.

- Top k models you found in cross validation.
- Different snapshots of the trained model.

http://cs231n.github.io/neural-networks-3/

Get the Most From Your Models

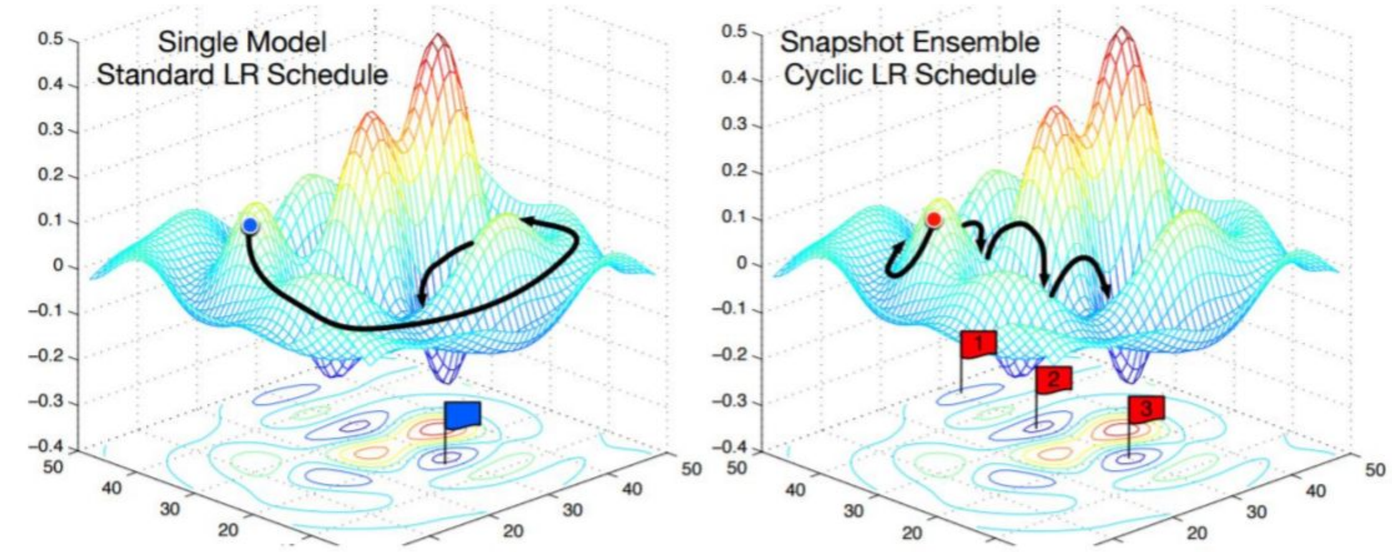
Model Ensembles

Instead of having a single trained model, you could generate a number of similar models and get a slightly better overall performance.

Final result is an averaged consensus over all models in your ensemble.

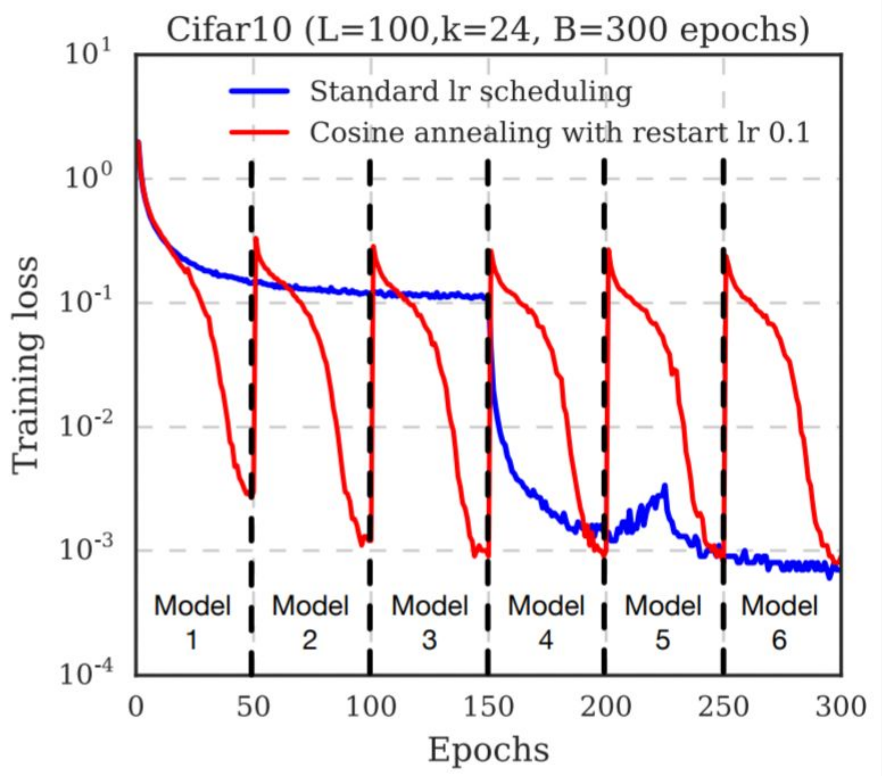
Could also take a weighted vote based on their validation accuracy.

An Optimization View of Model Ensembles



Loshchilov and Hutter, “SGDR: Stochastic gradient descent with restarts”, arXiv 2016
Huang et al, “Snapshot ensembles: train 1, get M for free”, ICLR 2017
Figures copyright Yixuan Li and Geoff Pleiss, 2017. Reproduced with permission.

Cyclic Learning Rate



Huang, Gao, et al. "Snapshot ensembles: Train 1, get m for free." *arXiv preprint arXiv:1704.00109* (2017).

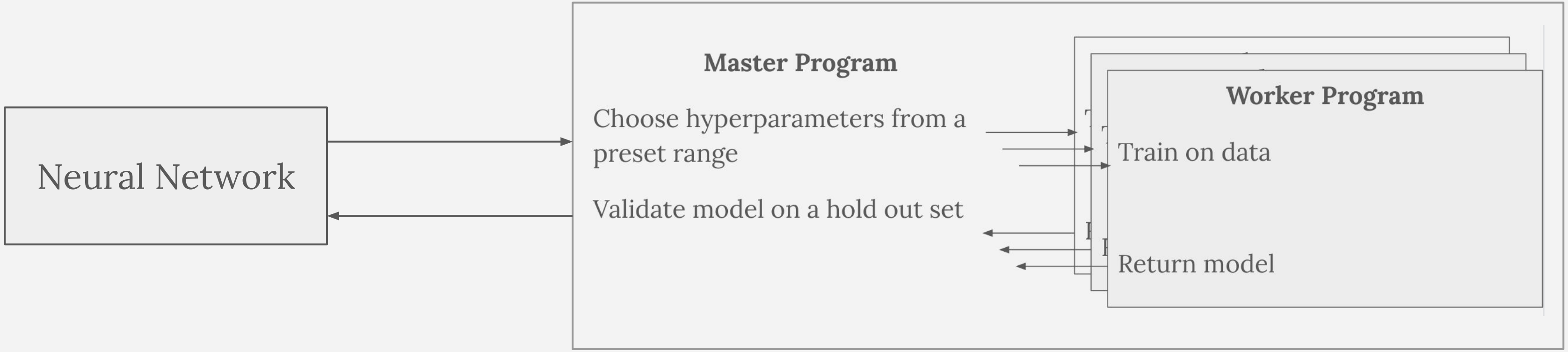
Today - Neural Network Architectures and Model Search

- Neural Network Architecture Design
- Hyperparameters and Network Sensitivity
- Optimize Hyperparameters Systematically
- **Automatic Model Search - AutoML**

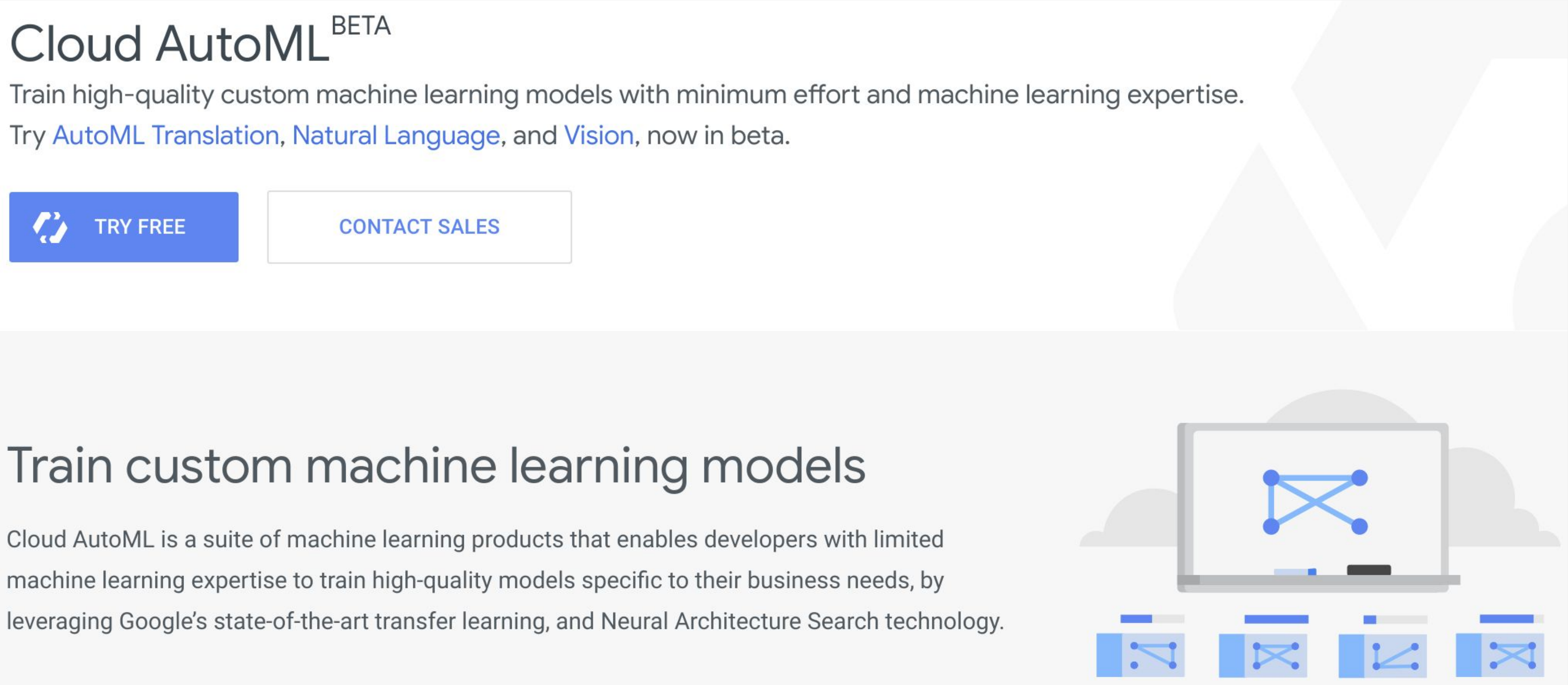
Automatic Model Search

The model search process is really another optimization problem.

Maybe we can train a neural network for that!


Neural Network
Master Program
Choose hyperparameters from a preset range
Validate model on a hold out set
Worker Program
Train on data
Return model

Automatic Model Search



Cloud AutoML$^{\text{BETA}}$

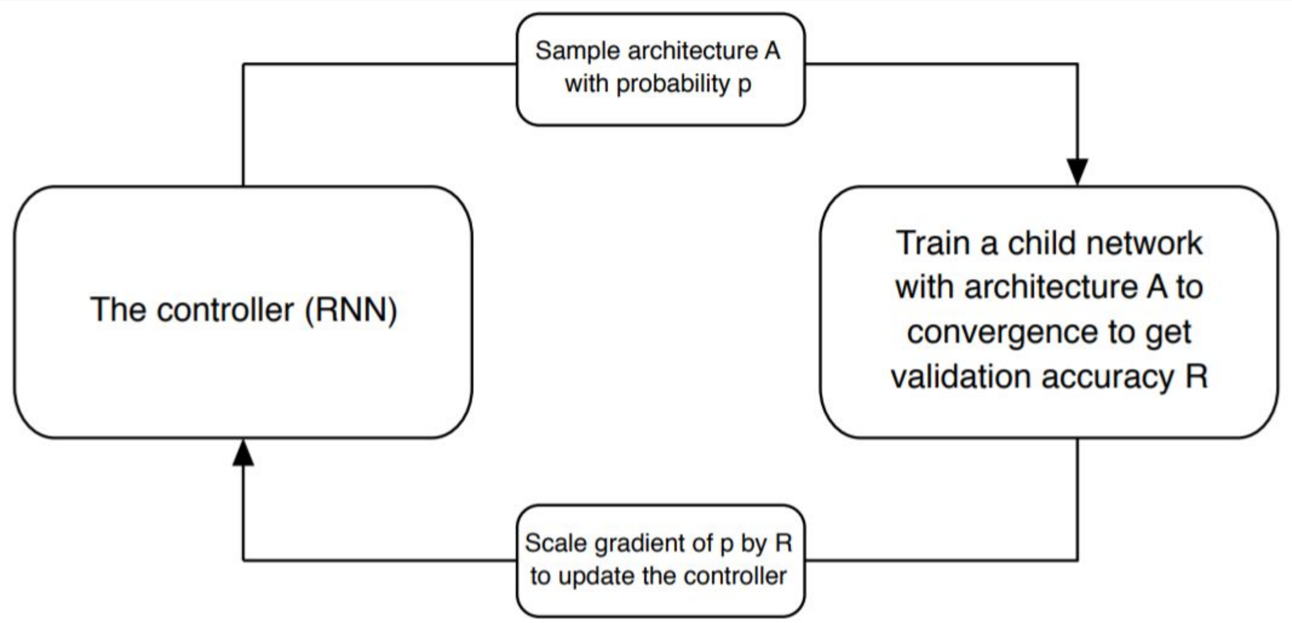
Train high-quality custom machine learning models with minimum effort and machine learning expertise.
Try AutoML Translation, Natural Language, and Vision, now in beta.

Train custom machine learning models

Cloud AutoML is a suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs, by leveraging Google's state-of-the-art transfer learning, and Neural Architecture Search technology.

https://cloud.google.com/automl/

Neural Architecture Search

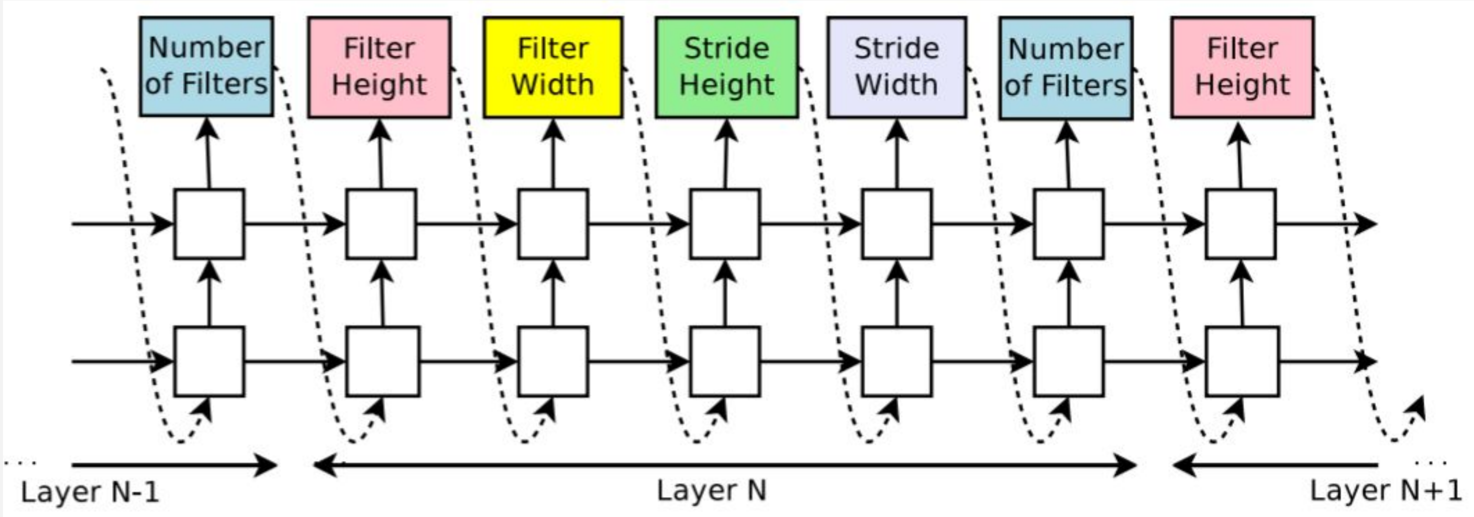


https://arxiv.org/abs/1611.01578
https://arxiv.org/pdf/1707.07012.pdf

Generate Model Descriptions

A Controller Recurrent Neural Network

At each layer, it predicts hyperparameters to construct a network unit.



Reinforcement Learning

Reward signal:

Validation accuracy

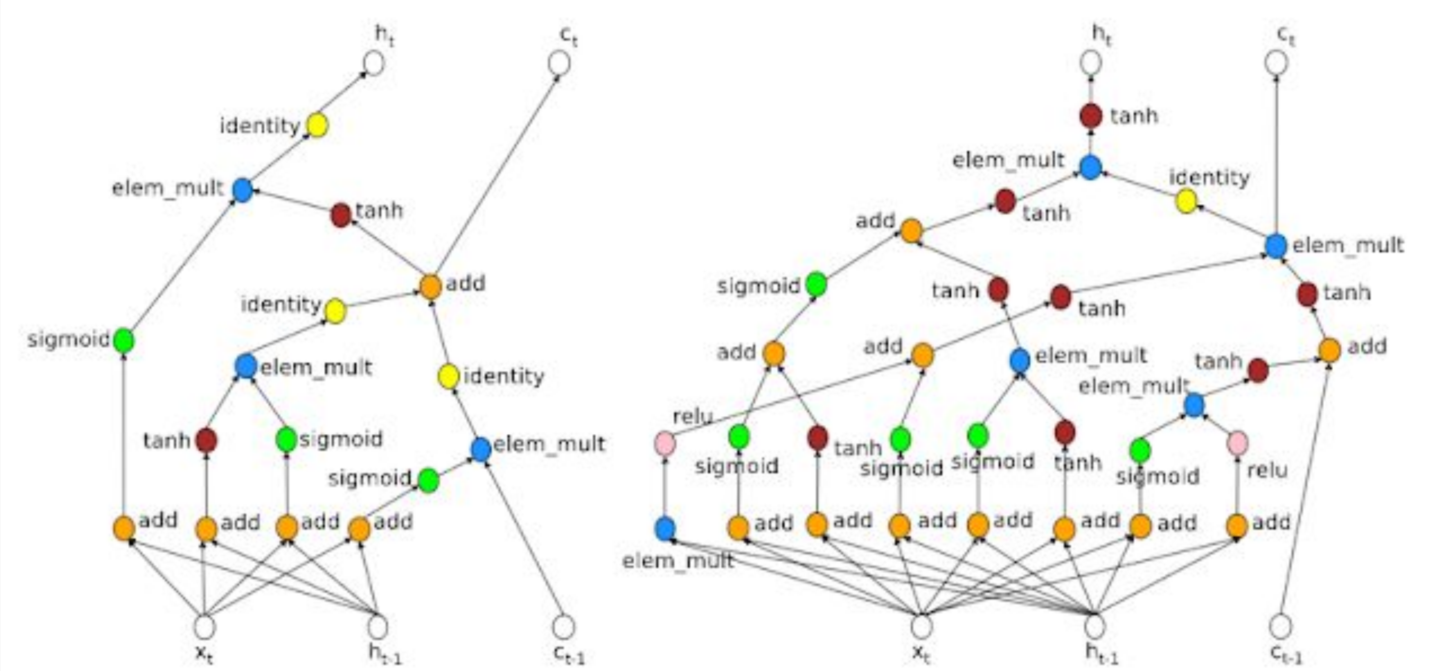
Why reinforcement learning in this task?

Supervised learning is not suitable because it doesn’t make sense to set an exact target model structure: we are searching for one!

You want to optimize the model structure given some high level ‘goodness’ measure.

Example of Found Structures

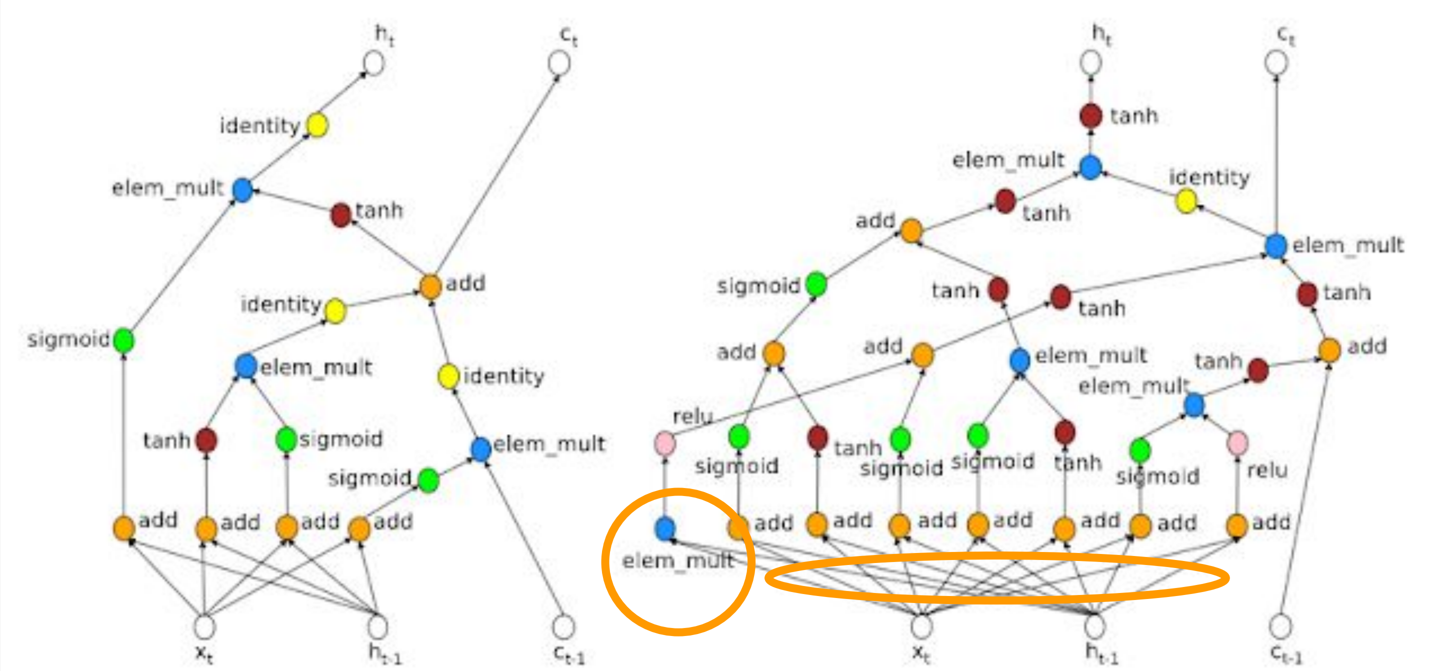
Human Proposed NAS Found



Example of Found Structures

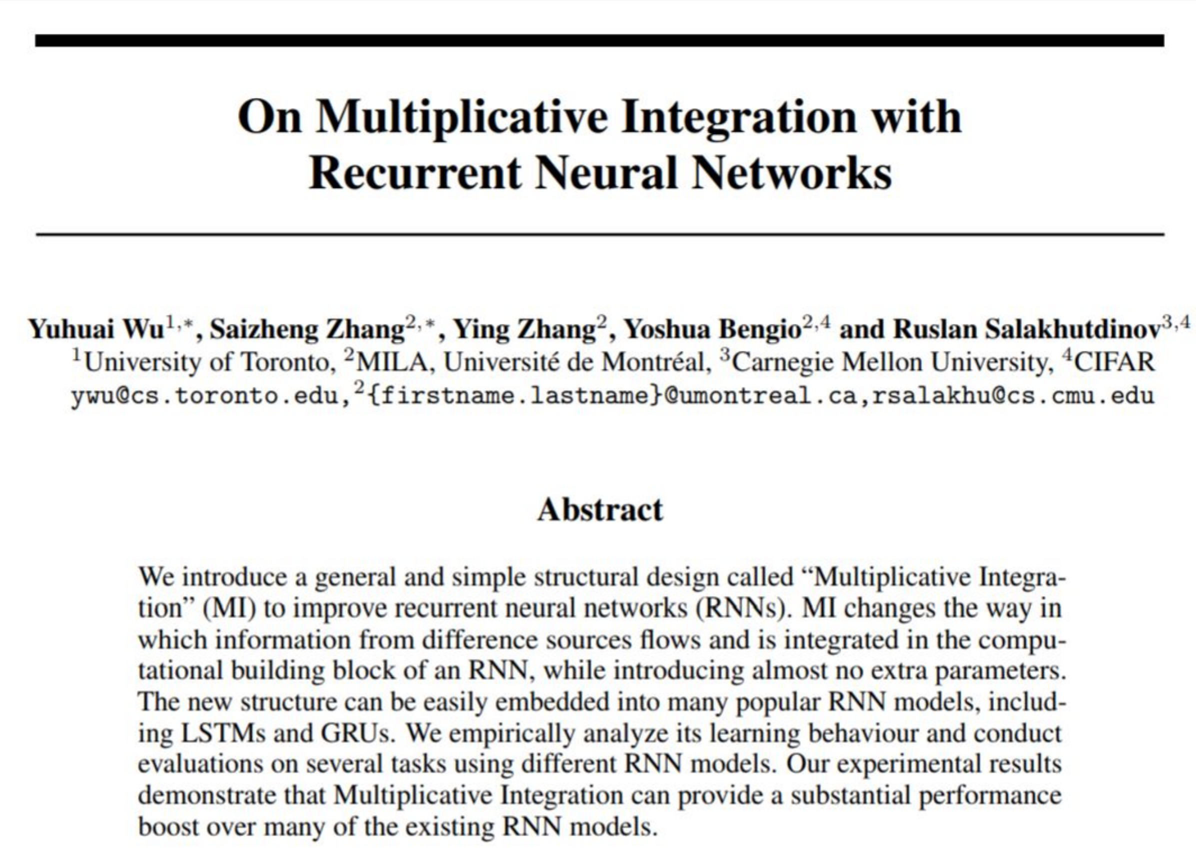
Human Proposed

NAS Found



New type of feature

A lot of connections

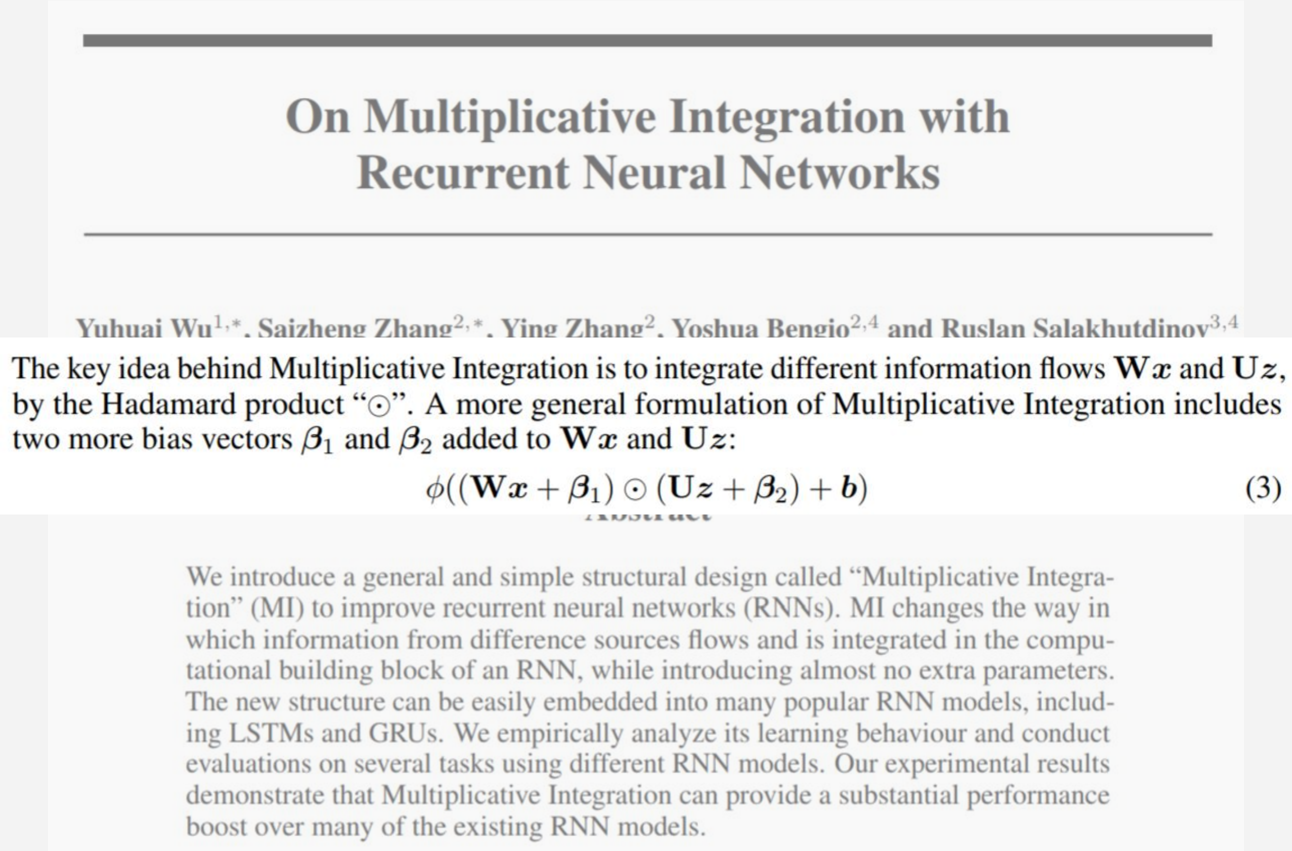


On Multiplicative Integration with Recurrent Neural Networks

Yuhuai Wu[1,*], Saizheng Zhang[2,*], Ying Zhang[2], Yoshua Bengio[2,4] and Ruslan Salakhutdinov[3,4]
[1]University of Toronto, [2]MILA, Université de Montréal, [3]Carnegie Mellon University, [4]CIFAR
ywu@cs.toronto.edu,[2]{firstname.lastname}@umontreal.ca,rsalakhu@cs.cmu.edu

Abstract

We introduce a general and simple structural design called "Multiplicative Integration" (MI) to improve recurrent neural networks (RNNs). MI changes the way in which information from difference sources flows and is integrated in the computational building block of an RNN, while introducing almost no extra parameters. The new structure can be easily embedded into many popular RNN models, including LSTMs and GRUs. We empirically analyze its learning behaviour and conduct evaluations on several tasks using different RNN models. Our experimental results demonstrate that Multiplicative Integration can provide a substantial performance boost over many of the existing RNN models.



On Multiplicative Integration with Recurrent Neural Networks

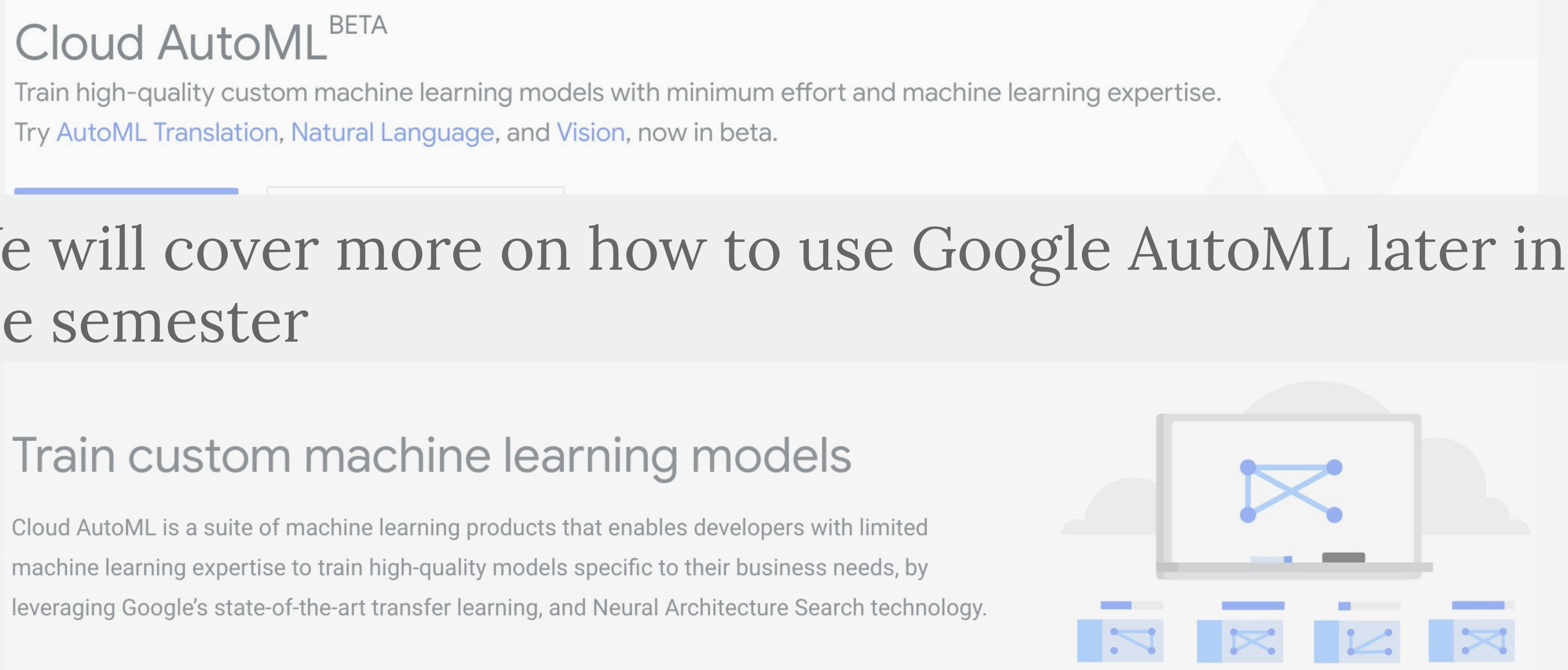
Yuhuai Wu[1,*], Saizheng Zhang[2,*], Ying Zhang[2], Yoshua Bengio[2,4] and Ruslan Salakhutdinov[3,4]

The key idea behind Multiplicative Integration is to integrate different information flows $\mathbf{W}\boldsymbol{x}$ and $\mathbf{U}\boldsymbol{z}$, by the Hadamard product "$\odot$". A more general formulation of Multiplicative Integration includes two more bias vectors $\boldsymbol{\beta}_1$ and $\boldsymbol{\beta}_2$ added to $\mathbf{W}\boldsymbol{x}$ and $\mathbf{U}\boldsymbol{z}$:

$$\phi((\mathbf{W}\boldsymbol{x} + \boldsymbol{\beta}_1) \odot (\mathbf{U}\boldsymbol{z} + \boldsymbol{\beta}_2) + \boldsymbol{b}) \qquad (3)$$

Abstract

We introduce a general and simple structural design called "Multiplicative Integration" (MI) to improve recurrent neural networks (RNNs). MI changes the way in which information from difference sources flows and is integrated in the computational building block of an RNN, while introducing almost no extra parameters. The new structure can be easily embedded into many popular RNN models, including LSTMs and GRUs. We empirically analyze its learning behaviour and conduct evaluations on several tasks using different RNN models. Our experimental results demonstrate that Multiplicative Integration can provide a substantial performance boost over many of the existing RNN models.



Cloud AutoML BETA

Train high-quality custom machine learning models with minimum effort and machine learning expertise.

Try AutoML Translation, Natural Language, and Vision, now in beta.

We will cover more on how to use Google AutoML later in the semester

Train custom machine learning models

Cloud AutoML is a suite of machine learning products that enables developers with limited machine learning expertise to train high-quality models specific to their business needs, by leveraging Google's state-of-the-art transfer learning, and Neural Architecture Search technology.

https://cloud.google.com/automl/

Summary

- Neural Network Architecture Design
- Hyperparameters and Network Sensitivity
- Optimize Hyperparameters Systematically
- Automatic Model Search - AutoML