Some logistics

- Video is posted shortly after class to forum thread
- You can join class remotely up to 2 times if you have to
- Please ask questions or add comments through forums
  - Questions with <*x*>+ likes will be asked by Rachel if she thinks it's of general interest, so be sure to 'like' anything you want to see addressed
  - If you aren't sure whether to ask – just ask anyway! There are no stupid questions 😊
- I'll be looking at questions after class too

Get to know the forum

Shift-r to reply; Ctrl-enter to send

Replies are instant: join the 'lesson 8 in-class' thread now!

You can watch and pin threads

Try creating a thread for your area of interest, for ongoing discussion

Feel free to experiment with ways to organize study groups, topics you're interested in, etc

Don't at-mention to encourage an answer, or the people you at-mention may turn off notifications! If you don't get an answer, try to rephrase

Study group

Room 153 @ USF (101 Howard St)

For in person participants only.

Interstate/international visitors priority if full

Generally someone is there at least 10a-4p

GitHub, Inc. [US] | https://github.com/fastai/fastai_docs/tree/master/dev_course/dl2

Search or jump to… /   Pull requests   Issues   Marketplace   Explore

fastai / fastai_docs   Watch

Code   Issues 0   Pull requests 0   Actions   Projects 0   Wiki   Insights

Branch: master   fastai_docs / dev_course / dl2 /   Create new

stas00 make strip

..	
exp	lesson updates
00_exports.ipynb	stripout
01_matmul.ipynb	make strip
02_fully_connected.ipynb	make strip
03_minibatch_training.ipynb	fix script

UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

fast.ai

Deep learning…

…from the foundations. Lesson 8.

"Deep Learning from the Foundations"

(aka *"Impractical Deep Learning for Coders"*)

- Implement much of fastai from **Foundations**
  - Basic matrix calculus; Training loops w callbacks; Custom optimizer; custom annealing; regularization; dataset, loader, and blocks
  - Lots of layers and architectures
  - Jupyter dev, testing and docs
- Read papers, and implement them. We'll even do some new research!
- Learn object detection, seq2seq/attention, Transformer/XL, CycleGAN, Audio…
- Performance: distributed training, JIT, CUDA/C++
- Implement some of fastai in Swift

Top down

- Context
- Motivation
- Results

Bottom up (with code)

- See the connections
- Customize
- Performance

fast.ai Embracing Swift for Deep Learning

Written: 06 Mar 2019 by *Jeremy Howard*

So, why are we embracing Swift at this time? Because Swift for TensorFlow is the first serious effort I've seen to incorporate differentiable programming deep in to the heart of a widely used language that is designed from the ground up for performance.

High Performance Numeric Programming with Swift: Explorations and Reflections

Written: 10 Jan 2019 by *Jeremy Howard*

Over the past few weeks I've been working on building some numeric programming libraries for [Swift](). But wait, isn't Swift just what iOS programmers use for building apps? Not any more! Nowadays Swift runs on Linux and Mac, and can be used for [web applications](), [command line tools](), and nearly anything else you can think of.

PyTorch Pros

- Get work done now!
- Great ecosystem
- Docs & tutorials

S4TF Pros

- Write everything in Swift
- See exactly what's happening
- Opportunities

PyTorch Cons

- Python's performance
- Python's types
- Mismatch with backend libs

S4TF Cons

- Minimal ecosystem
- Very little works
- Lots to learn

What do we mean by “from the foundations”?

Recreate: fastai*

…and much of PyTorch:
matrix multiply, torch.nn, torch.optim, Dataset, DataLoader

Python	Python stdlib	Non-data science modules	PyTorch array creation, RNG, indexer	fastai.datasets	matplotlib

* but we’ll make it *even better*!

But why?...

Really experiment

Understand it by creating it

Tweak everything

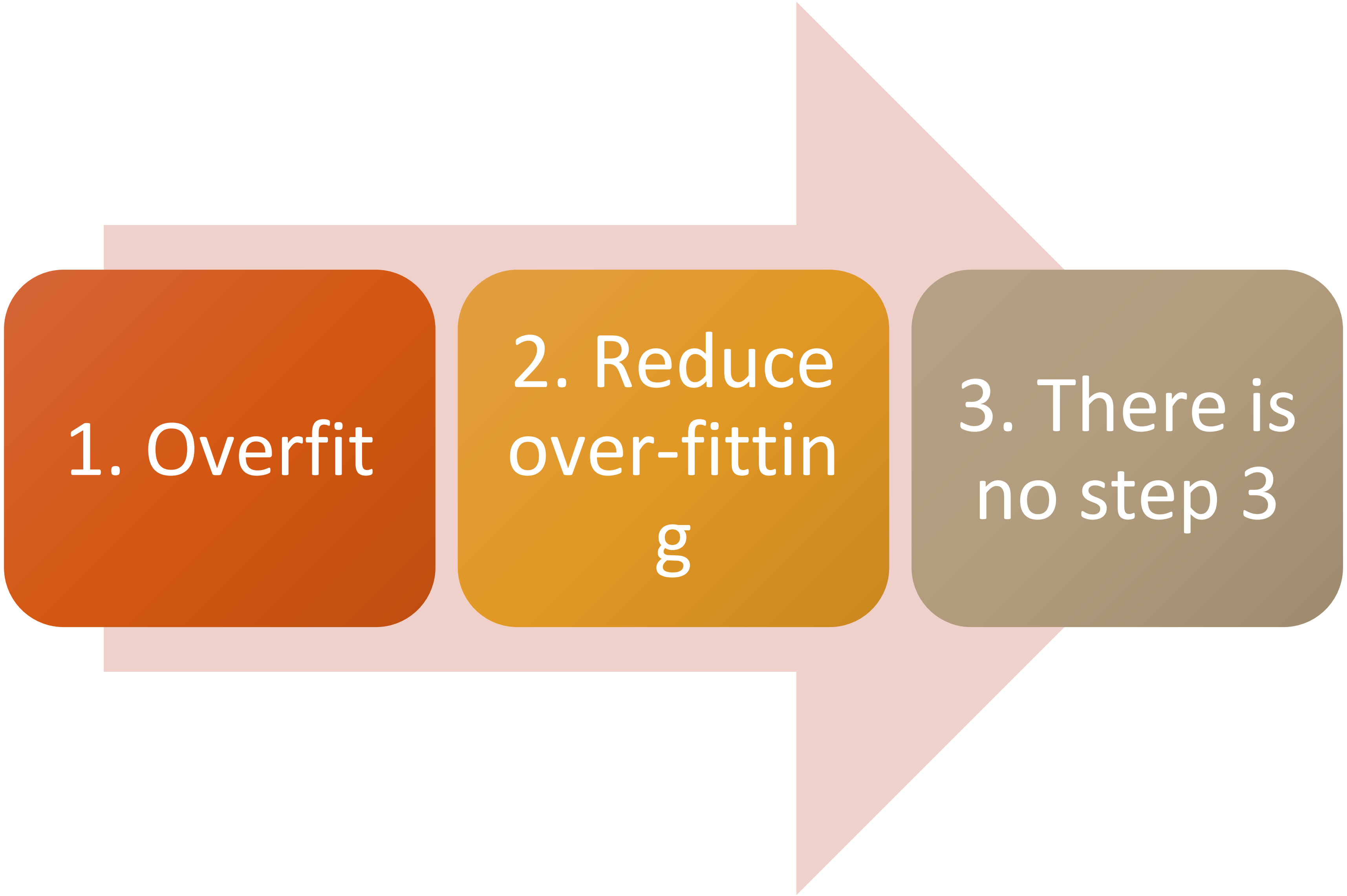
Contribute

Correlate papers with code

There are many opportunities for you in this class

- Your homework will be at the cutting edge
- There are few DL practitioners that know what you know now
- Experiment lots, especially in your area of expertise
- Much of what you find will have not be written about before
- Don’t wait to be perfect before you start communicating
- If you don’t have a blog, try medium.com

- Affine functions & non-linearities
- Parameters & activations
- Random init & transfer learning
- SGD; Momentum, Adam
- Convolutions
- Batch-norm
- Dropout
- Data augmentation
- Weight decay
- Res/dense blocks
- Image classification and regression
- Embeddings
- Continuous & Categorical Variables
- Collaborative filtering
- Language models; NLP classification
- Segmentation; U-net; GANs


1. Overfit
2. Reduce over-fitting
3. There is no step 3

Five steps to avoiding overfitting

- More data
- Data augmentation
- Generalizable architectures
- Regularization
- Reduce architecture complexity

It's time to start reading papers

Theorem 4.1. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$, and β_1, $\beta_2 \in [0, 1)$ satisfy $\frac{\beta_1^2}{\sqrt{\beta_2}} < 1$. Let $\alpha_t = \frac{\alpha}{\sqrt{t}}$ and $\beta_{1,t} = \beta_1 \lambda^{t-1}, \lambda \in (0, 1)$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\widehat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Our Theorem 4.1 implies when the data features are sparse and bounded gradients, the summation term can be much smaller than its upper bound $\sum_{i=1}^{d} \|g_{1:T,i}\|_2 << dG_\infty\sqrt{T}$ and $\sum_{i=1}^{d} \sqrt{T\widehat{v}_{T,i}} << dG_\infty\sqrt{T}$, in particular if the class of function and data features are in the form of section 1.2 in (Duchi et al., 2011). Their results for the expected value $\mathbb{E}[\sum_{i=1}^{d} \|g_{1:T,i}\|_2]$ also apply to Adam. In particular, the adaptive method, such as Adam and Adagrad, can achieve $O(\log d\sqrt{T})$, an improvement over $O(\sqrt{dT})$ for the non-adaptive method. Decaying $\beta_{1,t}$ towards zero is important in our theoretical analysis and also matches previous empirical findings, e.g. (Sutskever et al., 2013) suggests reducing the momentum coefficient in the end of training can improve convergence.

Finally, we can show the average regret of Adam converges,

Corollary 4.2. *Assume that the function f_t has bounded gradients, $\|\nabla f_t(\theta)\|_2 \leq G$, $\|\nabla f_t(\theta)\|_\infty \leq G_\infty$ for all $\theta \in R^d$ and distance between any θ_t generated by Adam is bounded, $\|\theta_n - \theta_m\|_2 \leq D$, $\|\theta_m - \theta_n\|_\infty \leq D_\infty$ for any $m, n \in \{1, ..., T\}$. Adam achieves the following guarantee, for all $T \geq 1$.*

$$\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}})$$

- Even familiar stuff looks complex in a paper!
- Papers are important for deep learning beyond the basics, but hard to read
- Google for a blog post describing the paper
- Learn to pronounce Greek letters

sebastianruder.com/optimizin

Adam

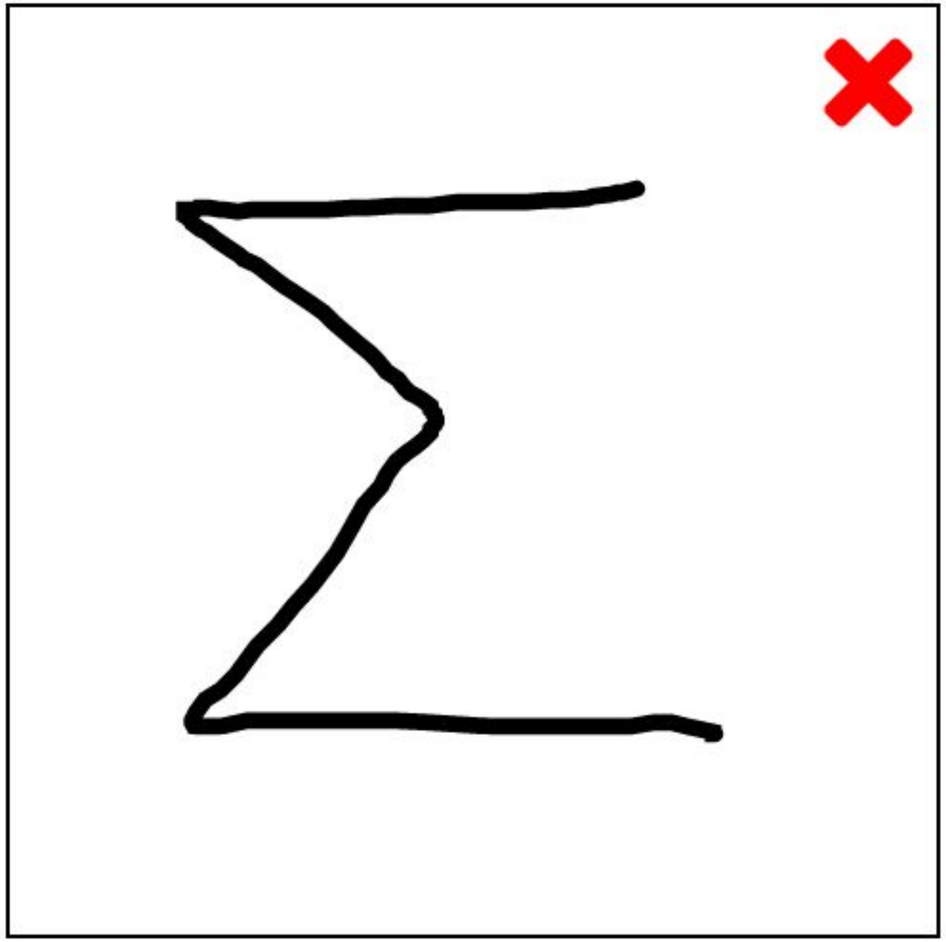
Adaptive Moment Estimation (Adam) [15] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

Symbols based on equality [edit]

Symbol in HTML	Symbol in TeX	Name Read as Category	Explanation	Examples
=	$=$ =	equality is equal to; equals everywhere	$x = y$ means x and y represent the same thing or value.	$2 = 2$ $1 + 1 = 2$ $36 - 5 = 31$
≠	$\ne$ \ne	inequality is not equal to; does not equal everywhere	$x \ne y$ means that x and y do not represent the same thing or value. (*The forms !=, /= or <> are generally used in programming languages where ease of typing and use of ASCII text is preferred.*)	$2 + 2 \ne 5$ $36 - 5 \ne 30$
≈	$\approx$ \approx	approximately equal is approximately equal to everywhere	*x* ≈ *y* means *x* is approximately equal to *y*. *This may also be written* ≃, ≅, ~, ♎ (Libra Symbol), *or* ≒.	π ≈ 3.14159
		isomorphism is isomorphic to	*G* ≈ *H* means that group *G* is isomorphic (structurally identical) to group *H*.	$Q_8 / C_2 \approx V$

Detexify

classify | symbols



Σ

Score: 0.06777255413017866

`\usepackage{ upgreek }`

`\Upsigma`

mathmode

Score: 0.09120779641372033

`\usepackage{ tipa }`

`\textrevyogh`

textmode

$\sum$

Score: 0.09998421642510583

`\sum`

mathmode

What do we mean by "from the foundations"?

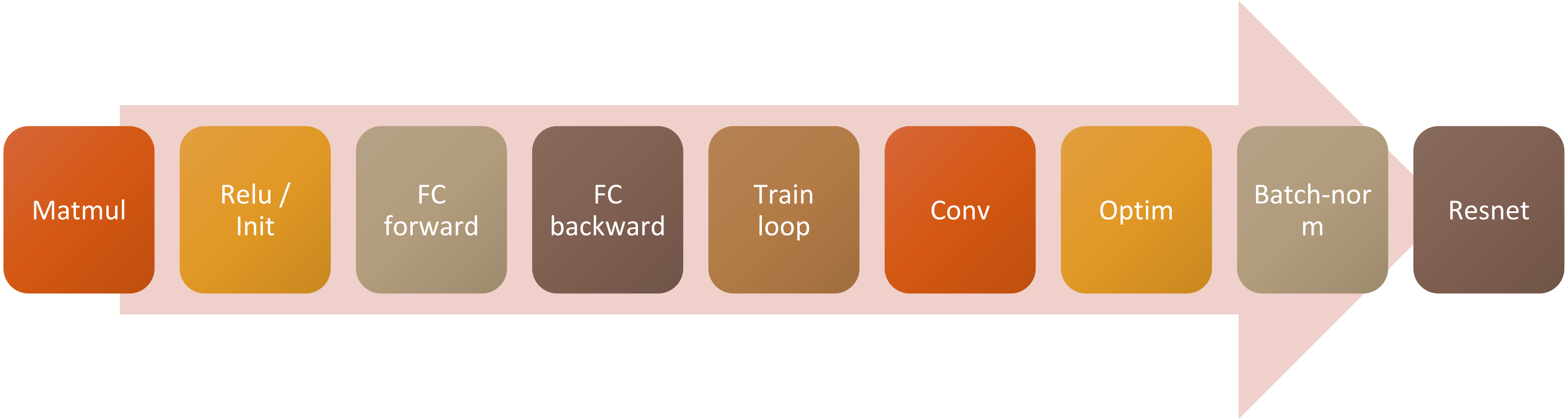
Recreate: fastai*

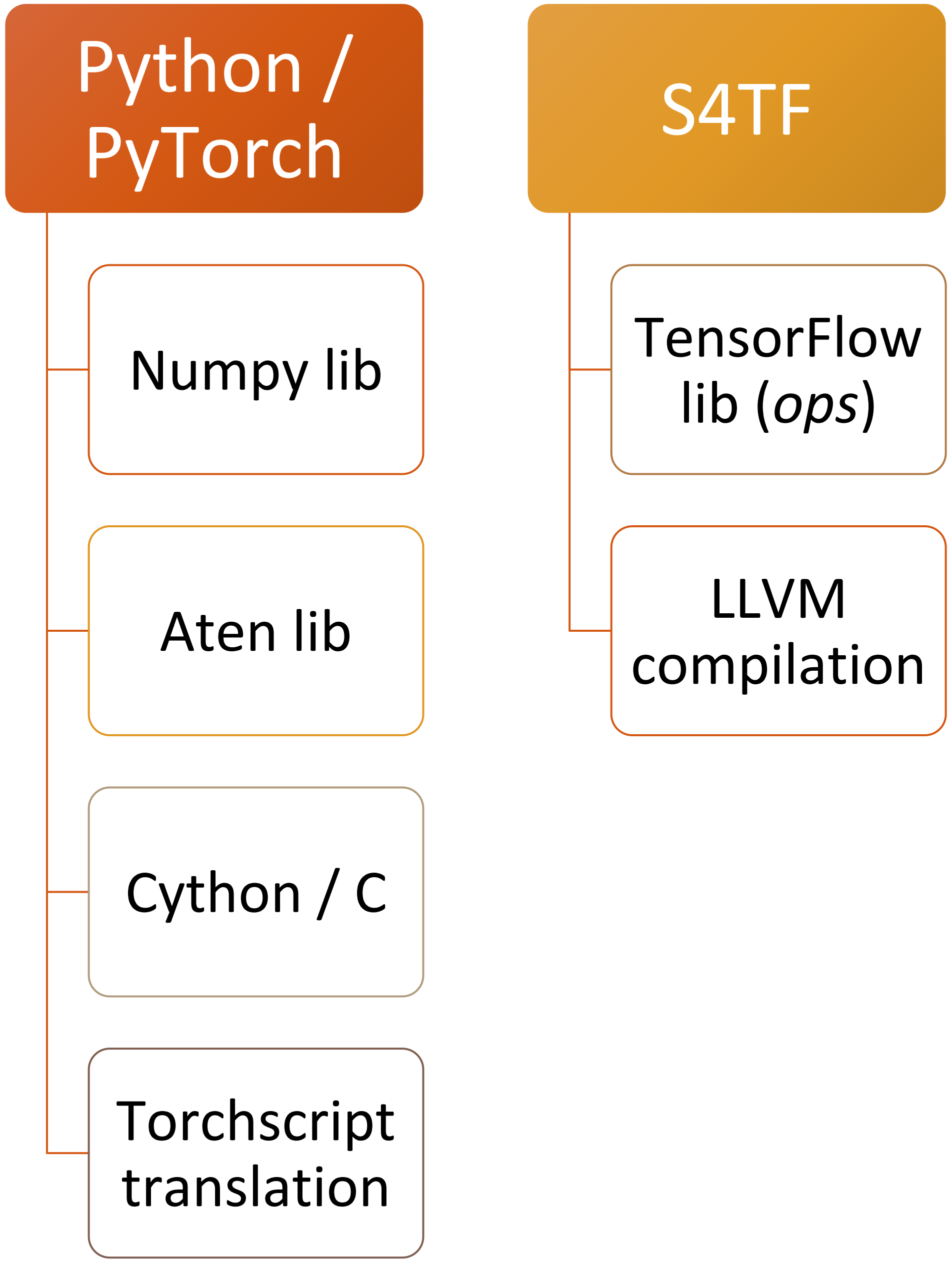
…and much of PyTorch:
matrix multiply, torch.nn, torch.optim, Dataset, DataLoader

Python	Python stdlib	Non-data science modules	PyTorch array creation, RNG, indexer	fastai.datasets	matplotlib

* but we'll make it *even better*!

Steps to a basic modern CNN model





Python / PyTorch
Numpy lib
Aten lib
Cython / C
Torchscript translation
S4TF
TensorFlow lib (*ops*)
LLVM compilation