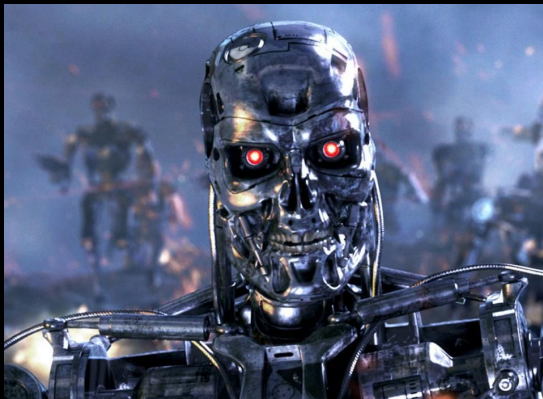


# Practical Methodology in Deep Learning

[shuchang.zhou@gmail.com](mailto:shuchang.zhou@gmail.com)

Jan. 22, 2019

# Deep Learning



What society think I do



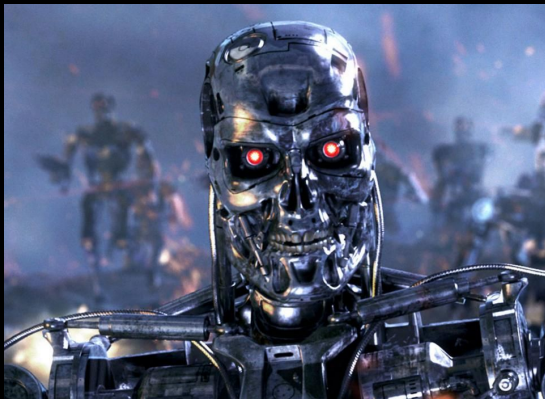
What my mom think I do



What I think I do

What I really do

# Deep Learning



What society think I do



What my mom think I do



What I think I do

CNN调参 每层200 包SOTA  
Caffe安装 20/次  
Tensorflow安装 15/次  
RNN调参 每层400 包SOTA  
GAN调参  
G网络800, D网络400

What I really do

# Debugging Learning Algorithm

# Regularized Supervised Learning

$$\min_{\theta} d(y, \hat{y}) + r(\hat{y})$$

$$\text{where } \hat{y} = f(X, \theta)$$

$d$  measures distance between ground truth and prediction

Regularizer:  $r$

$$d(y, \hat{y}) = -\log p(y|\hat{y})$$

$$r(\hat{y}) = -\log p(\hat{y})$$

$$\Rightarrow d(y, \hat{y}) + r(\hat{y}) = -\log p(y)$$

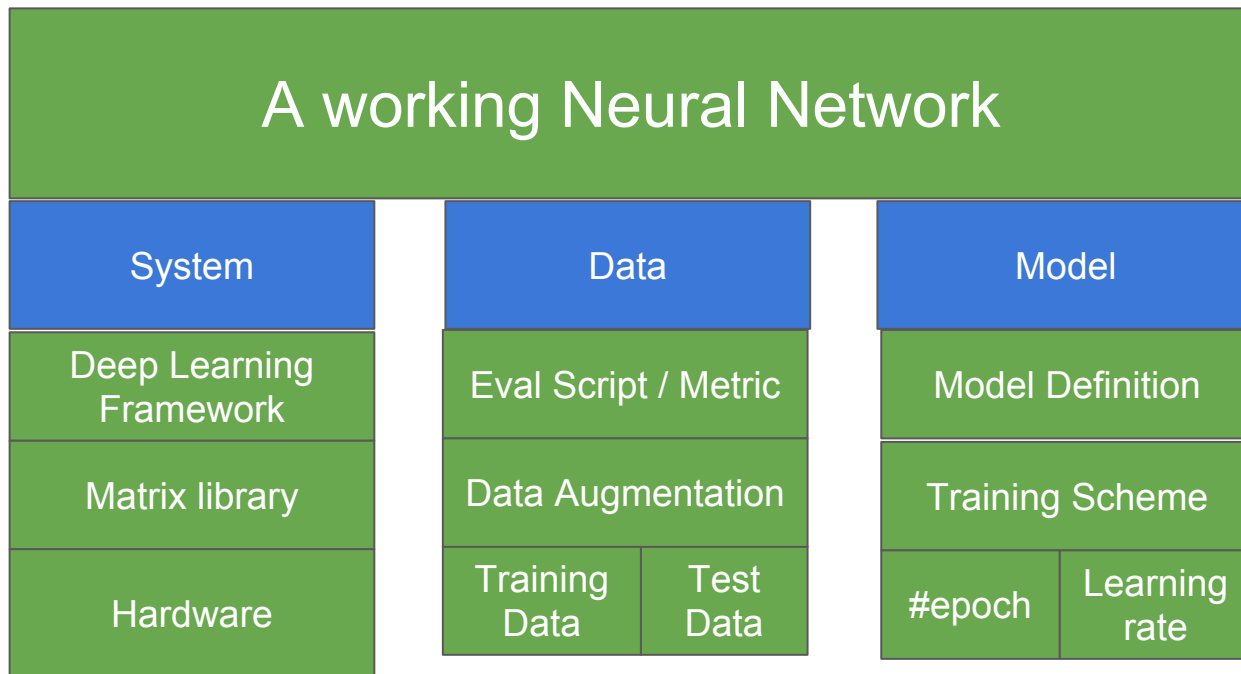
$d$  is conditional probability.

Regularizer  $r$  corresponds to prior.

$$C = d + r$$

$$\frac{d\theta}{dt} = -\frac{\partial C(\theta(t))}{\partial \theta} \Rightarrow \frac{dC(\theta)}{dt} = \frac{\partial C(\theta)}{\partial \theta} \frac{d\theta}{dt} = -\left(\frac{\partial C(\theta)}{\partial \theta}\right)^2 \leq 0$$

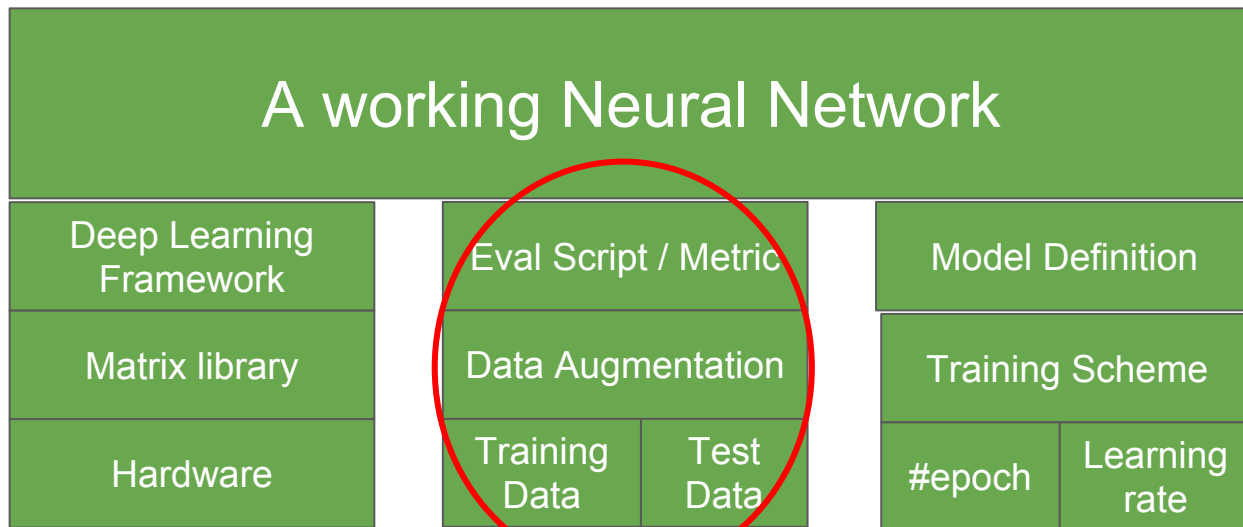
# Behind a working Neural Network



$$\min_{\theta} d(y, \hat{y}) + r(\hat{y})$$

where  $\hat{y} = f(X, \theta)$

# Behind a working Neural Network



Data bugs

$$\min_{\theta} d(y, \hat{y}) + r(\hat{y})$$

where  $\hat{y} = f(X, \theta)$

# Data bugs: improper input

- A neural network is to some extent robust to such distortions
  - Hard to spot subtle error
  - Need visualization
    - Listing the top errors
    - View random samples



Bad Aspect Ratio

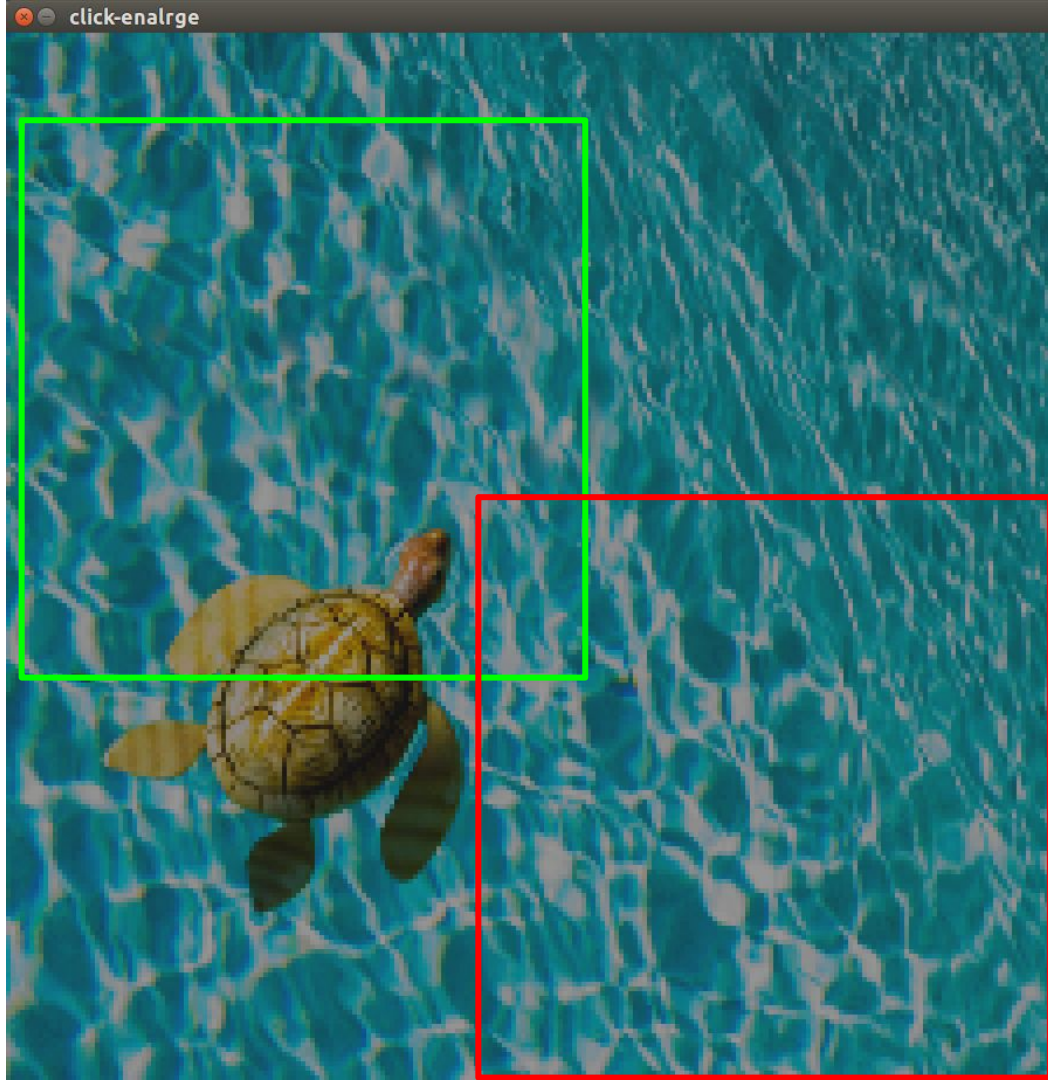


RGB or BGR?



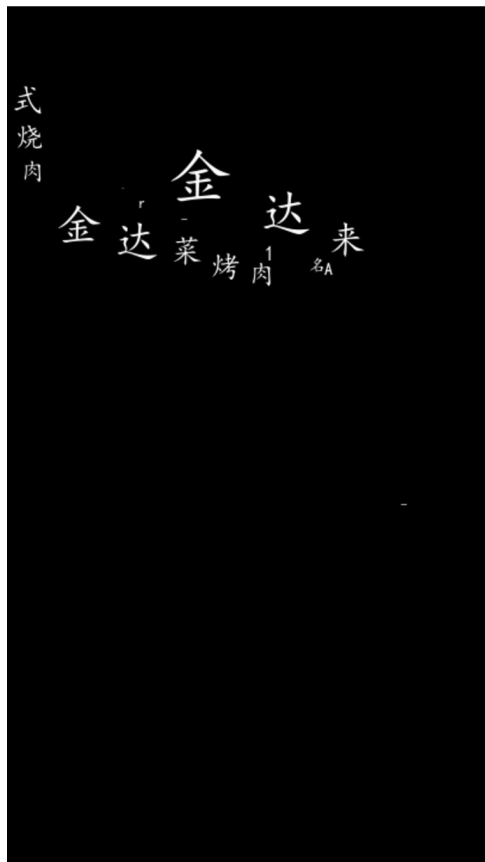
# Augmentation: random cropping

- Tradeoff between augmentation strength and augmented data quality
  - Too strong augmentation make data invalid
  - Too weak augmentation degrades generalization ability
- Rule-of-thumb
  - The bad augmentation percentage should not exceed prediction error rates.



# Need label be also transformed when augmenting?

For heatmaps, should also apply geometric transform. But should there also be blurring?



# Data bug: what the classifier really is



No

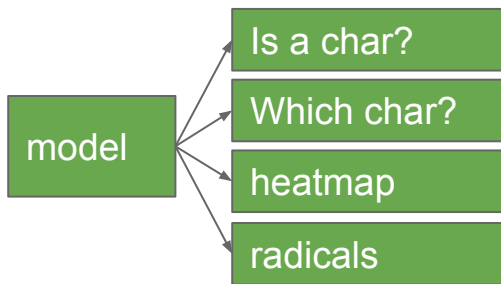
Tank classifier,  
or weather  
classifier?



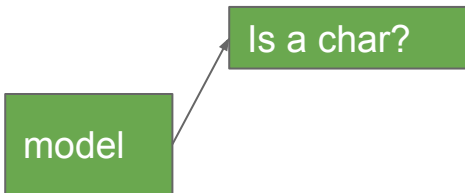
Yes

# Data bug: what the classifier really is

- Solution: multi-task learning
  - Can use additional supervision signal for training, but can omit when inference.



Train time

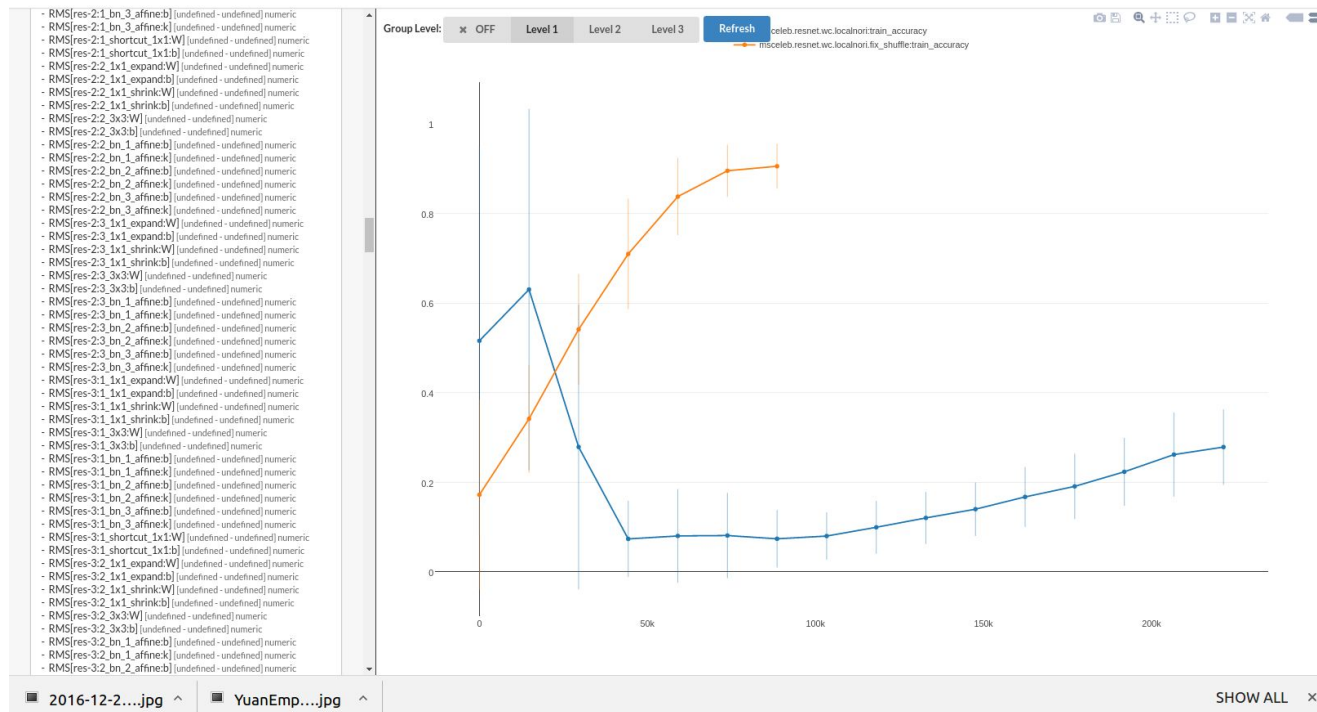


Inference time, no extra computation

# Data bug: sample with replacement is bad



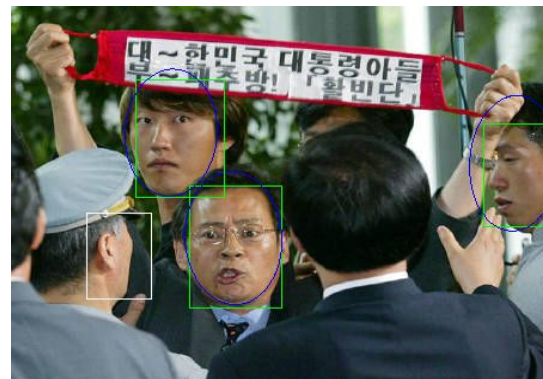
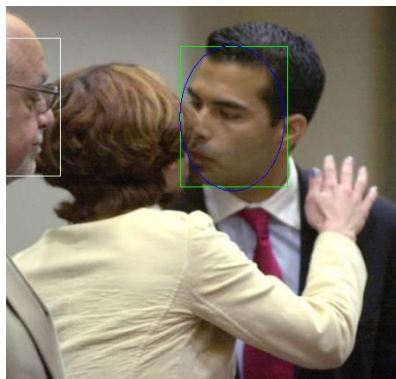
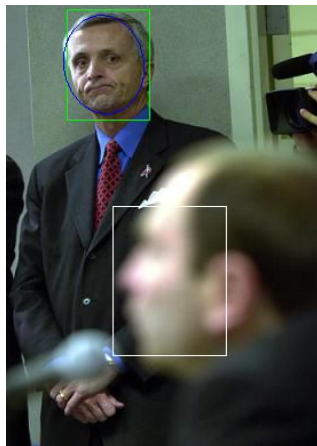
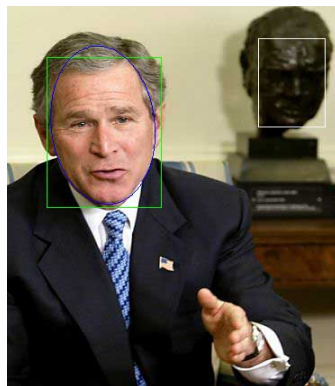
## Brain++ Monitor



***np.random.choice*** is  
with replacement by  
default.



# Eval metric can be wrong



Check the misclassifications



# Data checklist

- Training Data

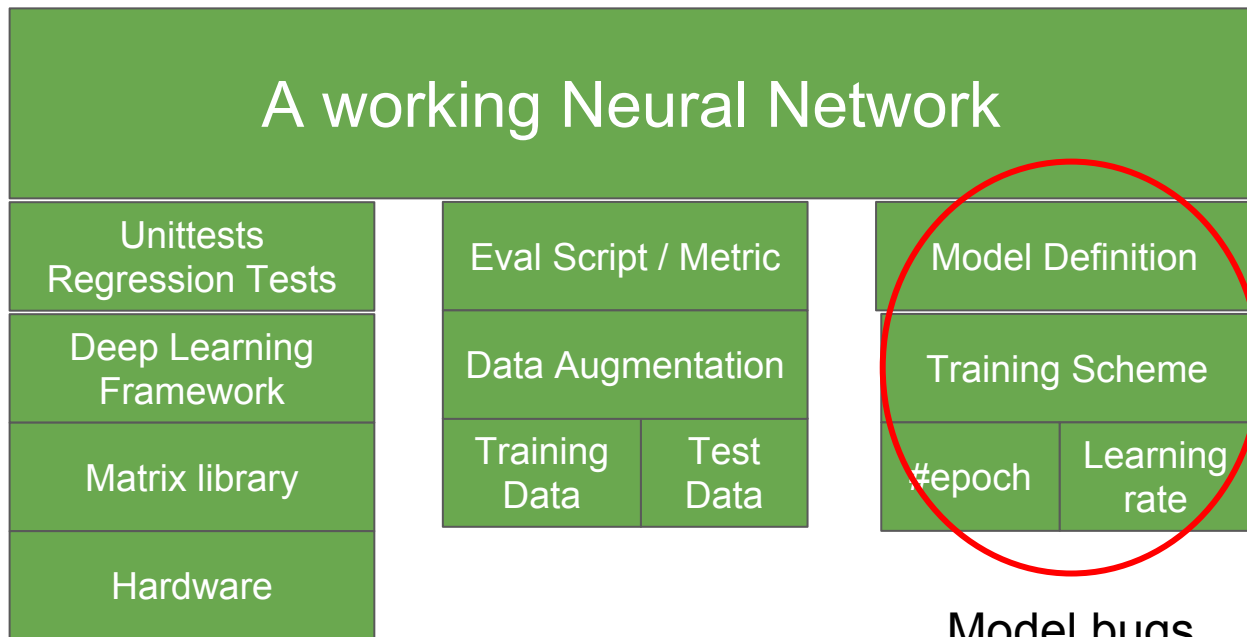
- Visualize the images and check the classes
  - Can a human learn the task?
  - De-duped and shuffled?
- Visualizing the augmented images
  - Augmentation too strong?
  - Need label to also transform?

- Test Data

- Drawn from the same distribution as Training data?
- Is it really segregated from Train?
  - Simply different images may be insufficient.
    - For face recognition, need be images of different persons
- Properly de-duped?



# Behind a working Neural Network



$$\min_{\theta} d(y, \hat{y}) + r(\hat{y})$$

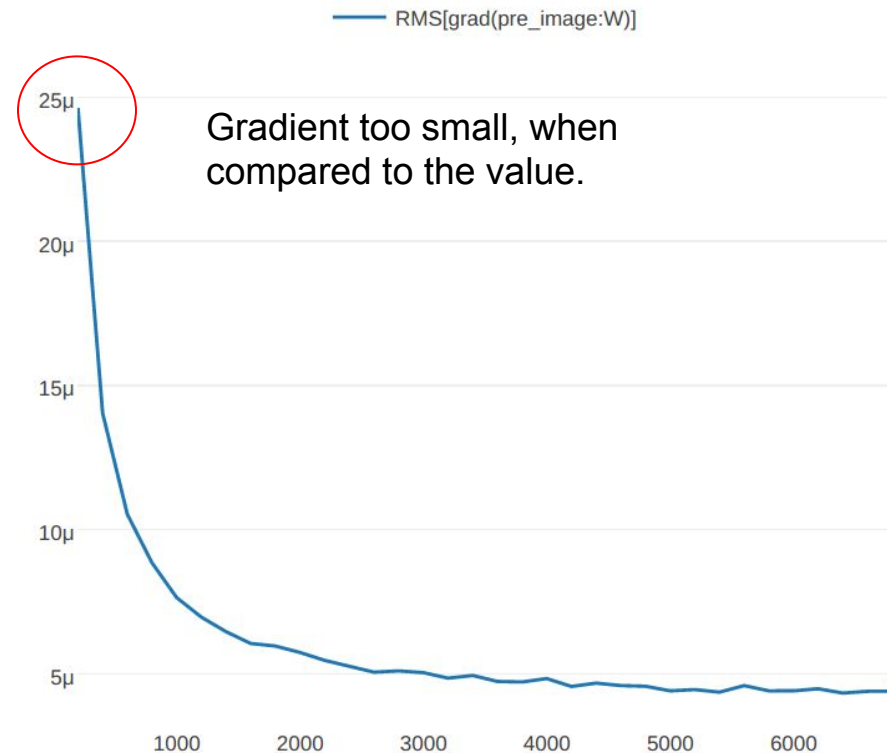
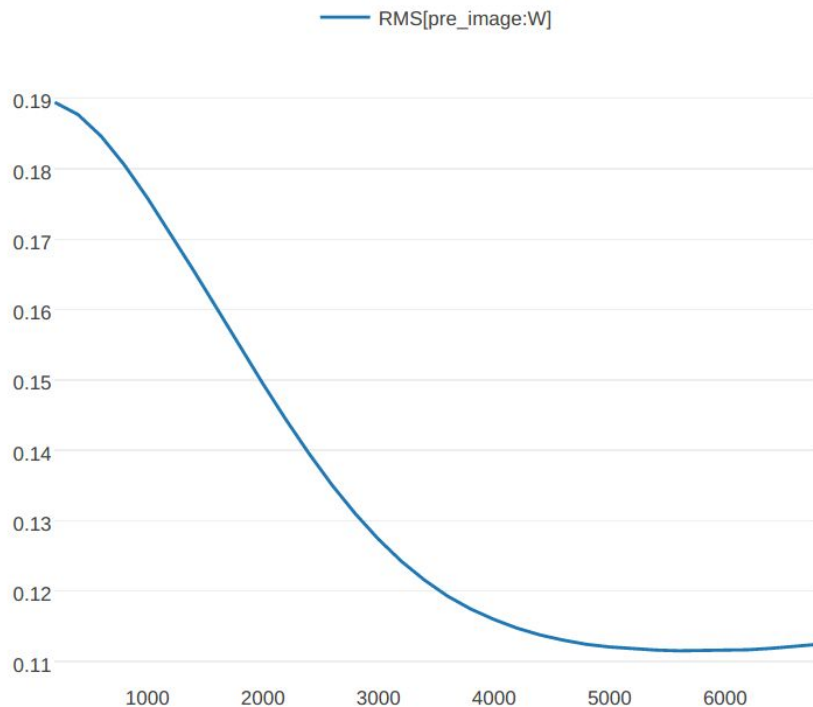
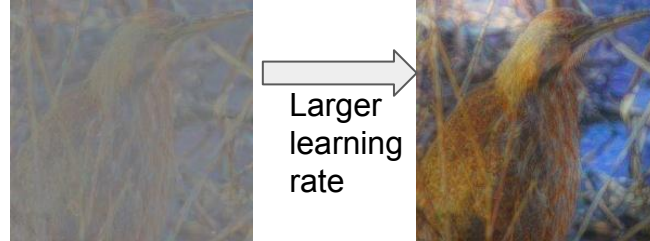
where  $\hat{y} = f(X, \theta)$

# Model bug: weight value range

- Symptom
  - Binarized model converges slowly on MNIST
    - Reference implementation has validation error rate 3.53% after 3 epochs
    - Our implementation only gets 14% after 58 epochs
- Debug Process
  - Read the logs, find the training misclassify also as high as 15%.
    - The model is underfitting
  - A line of control experiments to test the differences between ours and reference
- Conclusion
  - Need to change the value range of quantized weights from  $\{0, 1\}$  to  $\{-1, 1\}$ 
    - Gets 2.9% error rate after 10 epochs

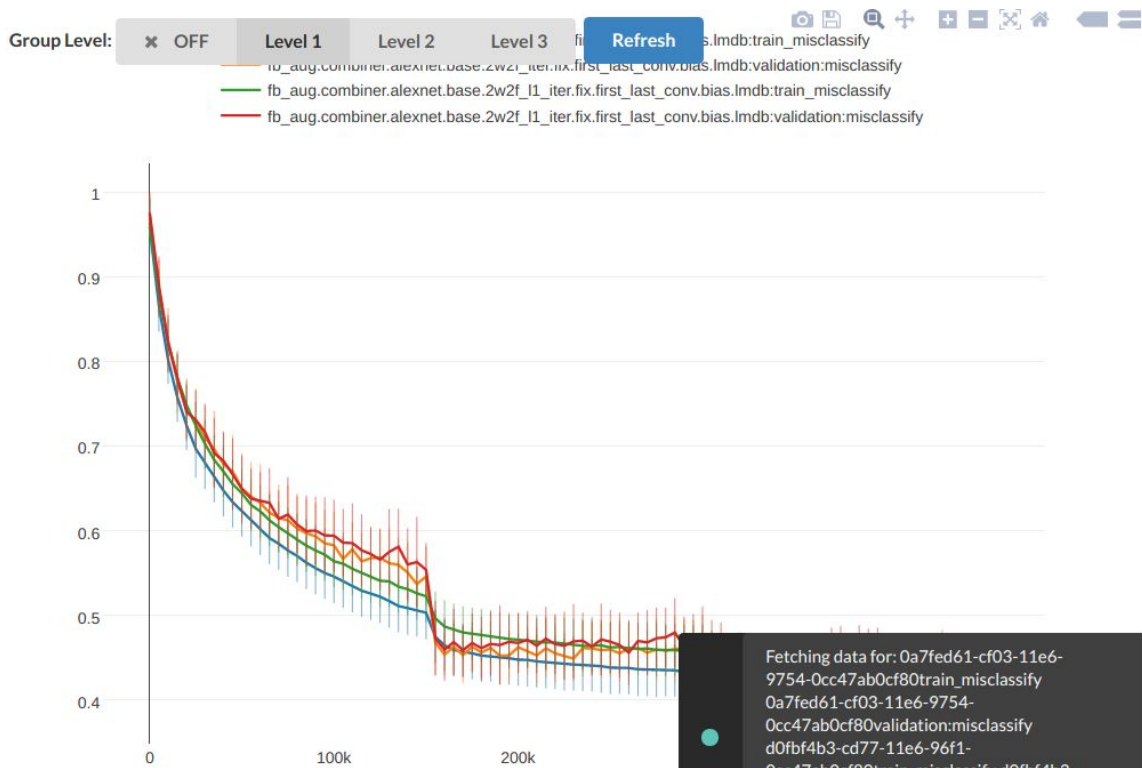
Ad: visit <http://dorefa.net> for more.

# Learning rate tweaking



# Diagnostics for debugging learning algorithms

- Training error near zero?
- Loss normal?
- Parameter scale normal?
- Gradient scale normal?
- Activation scale normal?
- Validation error near zero?



# Document the trials and ensure reproducible

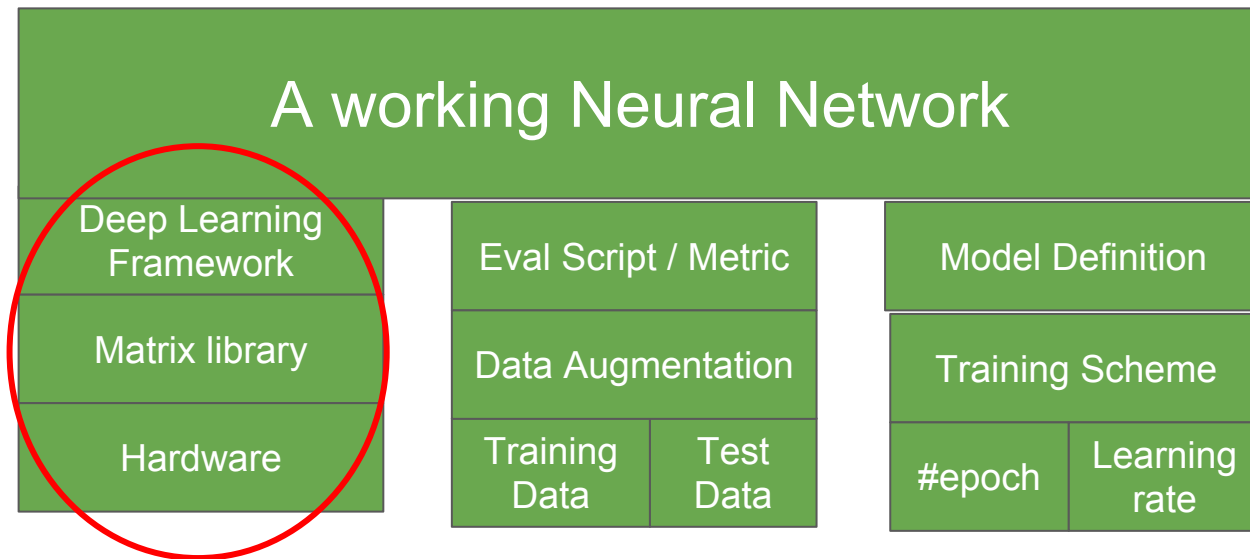
- Global Tree Naming

- Offers ablative analysis
- example: 中浙优8号
  - 中稻, 浙江产, 优, 8号
- Specialized names: 隆平稻
  - reserved for exceptional good ones
  - serve to shorten name

- svhn.quarter\_fc.no\_epsilon.no\_dupe.shuffled.lr\_1e-3.adam

- SVHN
- quarter\_fc: FC only has quarter #channel
- no\_epsilon + no\_dupe: remove epsilon and dupes in labels before matching
- Data are shuffled
- Learning rate 1e-3
- ADAM

# Behind a working Neural Network



System bugs

$$\min_{\theta} d(y, \hat{y}) + r(\hat{y})$$

where  $\hat{y} = f(X, \theta)$

# System bugs

- Source
  - Problematic Optimizations for acceleration
    - Especially the branches for different architectures and input shapes.
  - Failing hardware
- Solution
  - Numeric unit tests, especially for gradients
  - Known-to-work NN, like MNIST/SVHN
  - Falling back to the most mature branch (X86) and turning off optimizations
  - A sense of *what-should-work*

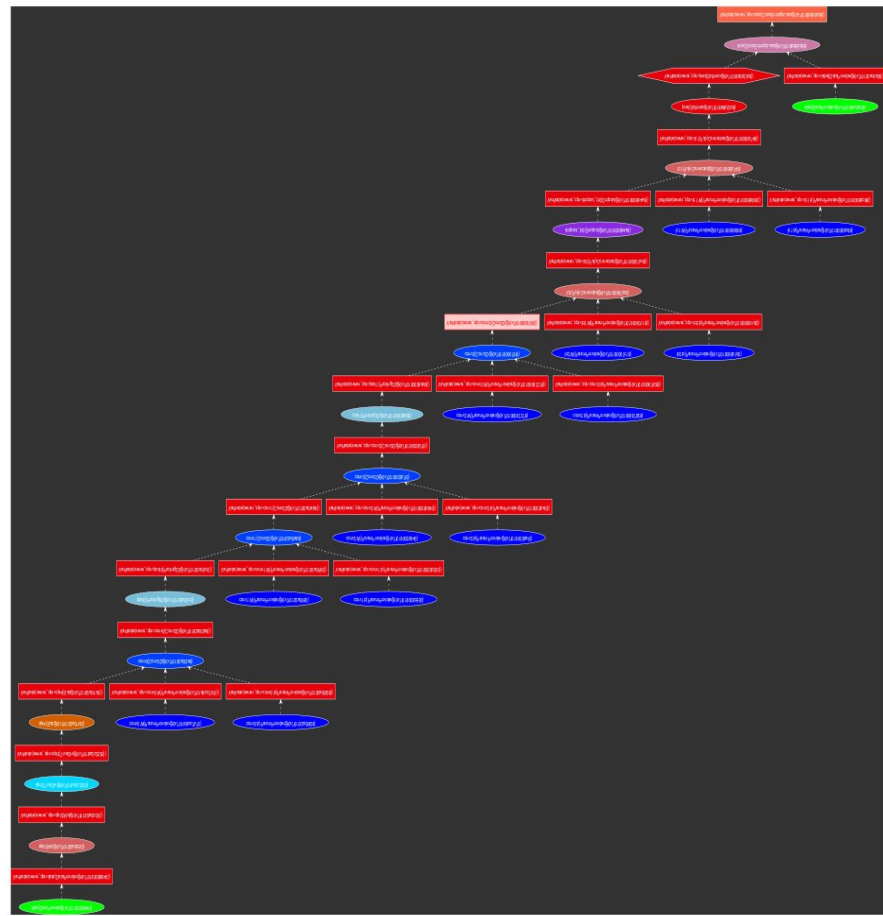
Gaming graphic card producing wrong numbers



```
[r:0,c:1,out] 10 13:42:41[mgb] ERR caught exception in async worker 'comp_node dispatch:gpu0:0'; what(): var sanity check failed: var: {id:2851180, layout:{1(1),1(1)}, Float32, owner:dimshuffle(fc2 bn affine:k)[94764]:dup{Dimshuffle}, name:dimshuffle(fc2 bn affine:k)[94764]:dup, slot:0, gpu0:0, s} (checksum: expect={checksum:0xbfd281e7, last int:-1076723225, last float:-1.64459} got={checksum:0xbfd26a62, last int:-1076729246, last float:-1.64387}); receiver: MUL(dimshuffle[94764],POW[2645135])[2645139]:dup{Elemwise}(2854179); you can set MGB DEBUG VAR SANITY CHECK LOG=2851180 to get more details; pass=0
[r:0,c:1,out] bp:/opt/megdl/MegBrain/v5.14.2/ mgb.so{2e9158,28a241,319b41,319dd5}
```

# System bug example: model output different

- Symptom
  - The prediction of model is different on CPU and GPU
- Debug process
  - Triage through the operator tree for the node that has different outputs
- Conclusion
  - The OpenCV resize has several optimizations that make it different from the definition





# Speeding up Development

# Development Flow

- Define the problem
  - Have a benchmark and a numeric metric
  - An intuitive end-to-end demo
- Iteratively refine pipeline (trial-and-error)
  - Analyze and understand the numbers and figures
    - Diagnostics for debugging learning algorithms.
    - Error analysis and ablation analysis.
  - Periodically optimize for the speed, make the method more practical
  - Document the trials (success and failures) and ensure reproducibility

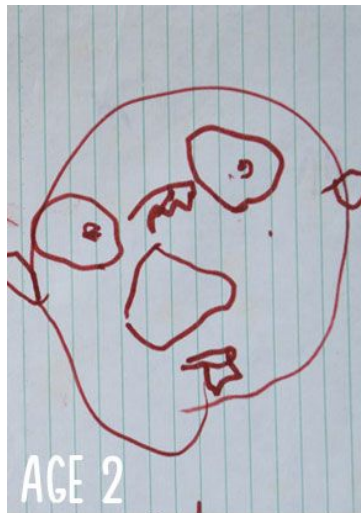
# Have a benchmark and a numeric metric

- OCR
  - Text positions and content
    - Deal with duplications
    - Single char, or word?
    - Which blur texts to ignore?



# Evolution of a pipeline

- Get a working pipeline first, before optimizing components.
- Pipeline should have less #hyper-parameters.

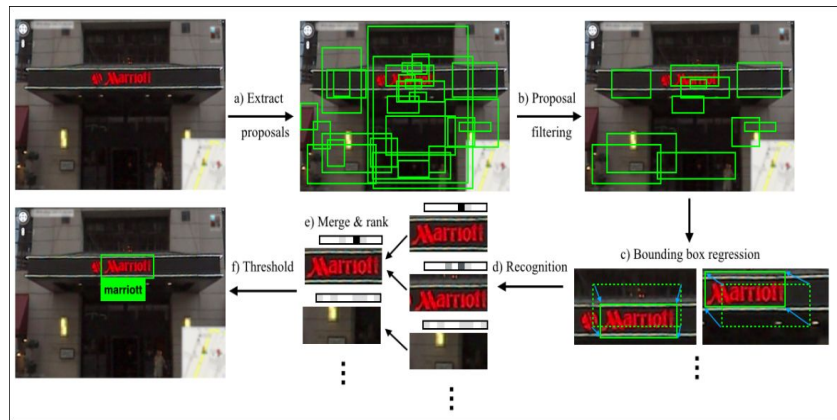


<https://www.youtube.com/watch?v=o5asMTdhmvA>

# Case study: OCR pipeline

- Merge these
  - Proposal
  - Filtering
  - Bounding box regression

Reading Text in the Wild with Convolutional Neural Networks, 2014



EAST: An Efficient and Accurate Scene Text Detector, 2017

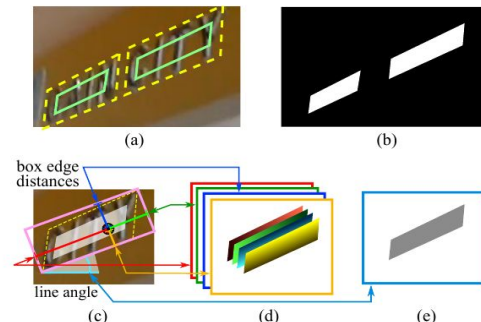
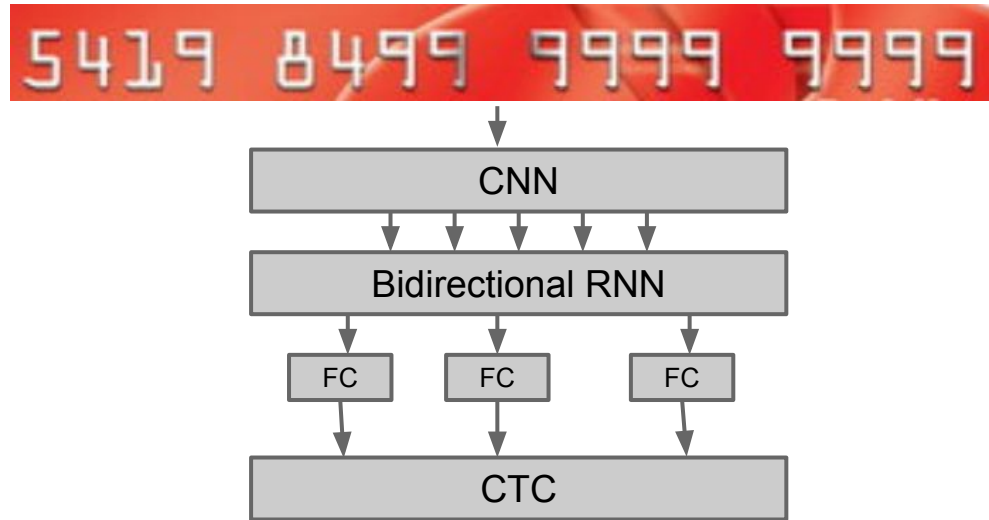
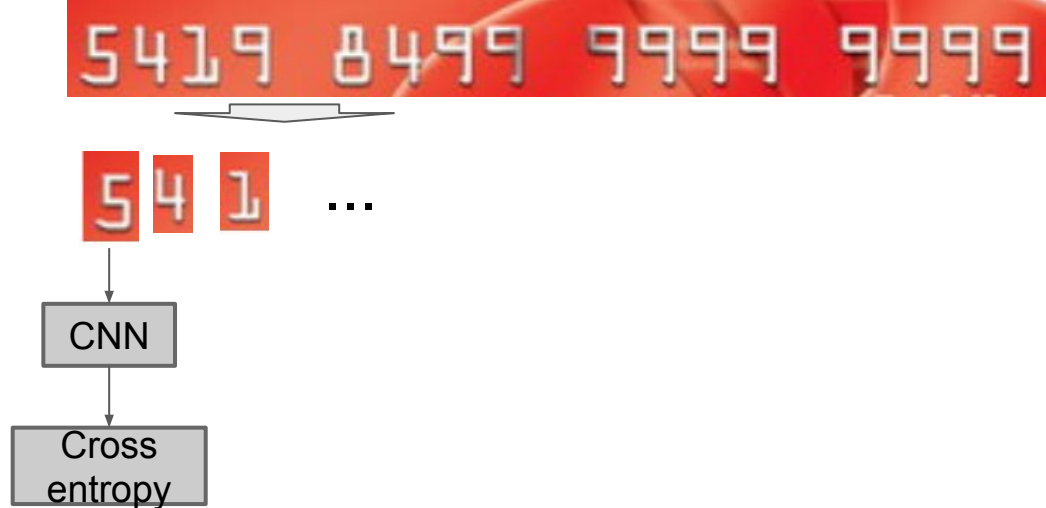


Figure 4. Label generation process: (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map; (c) RBOX geometry map generation; (d) 4 channels of distances of each pixel to the rectangle boundaries; (e) Text rotation angle.



# Pipeline design

- Prefer end-to-end
  - If no proper metrics can be defined
- Prefer staged
  - If near-perfect accuracy
  - Allows breaking up work
    - Need define protocols

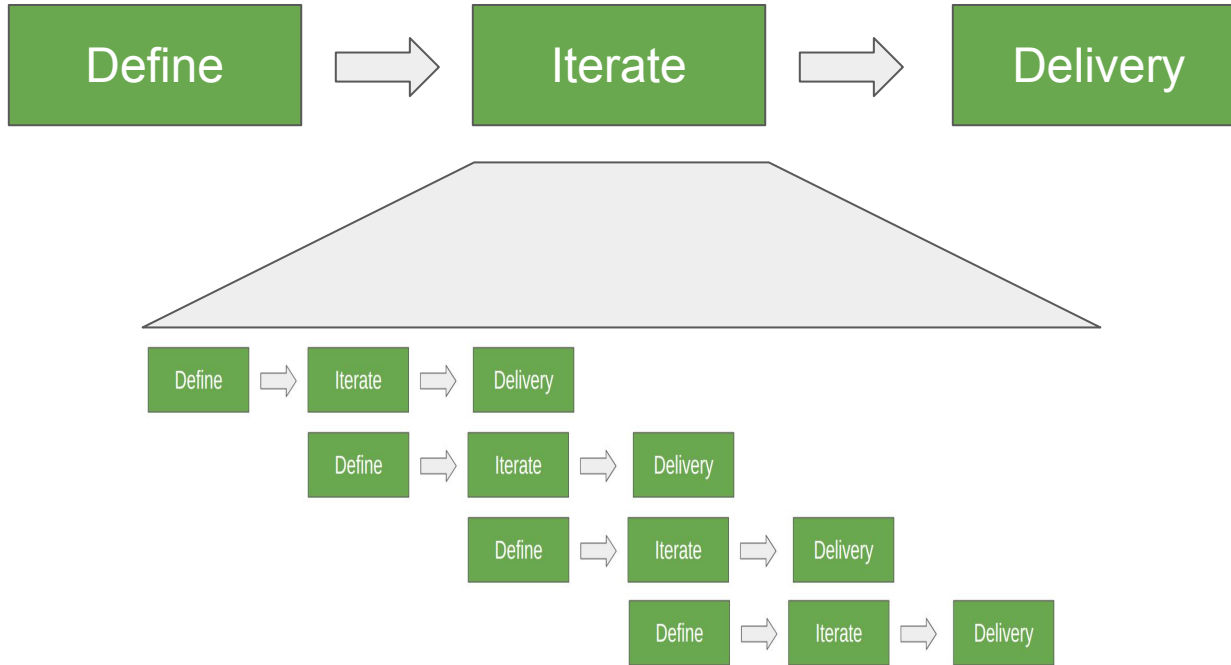


# Periodically optimize for the speed





# Pipeline of Trying Out Ideas



# Periodically optimize for the speed

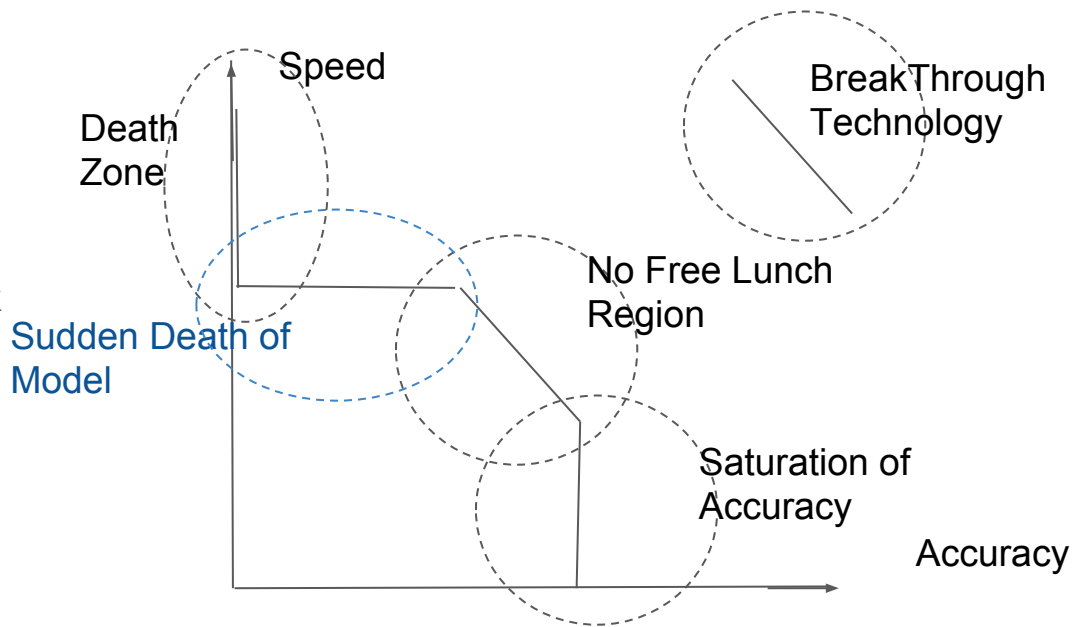
- Check list
  - Can we use less data?
  - Can we use smaller model?
  - Multi-card/multi-machine training?
  - Can augmentation speed be improved?
  - Can we parallelize the eval process?
  - **Can we automate the process?**

# Rule-of-thumb for designing a model

- Best reuse an existing model, or part of it.
  - Models pre-trained on large datasets are powerful.
- **$\log_2 \text{input\_image\_size} - 2$**  down-samplings
  - Input image should be large enough for human to judge
- Determine #channel from computation budget
- If translation invariant, then use more convolutions.
- If too few parameters
  - Fully-connected or Locally-connected

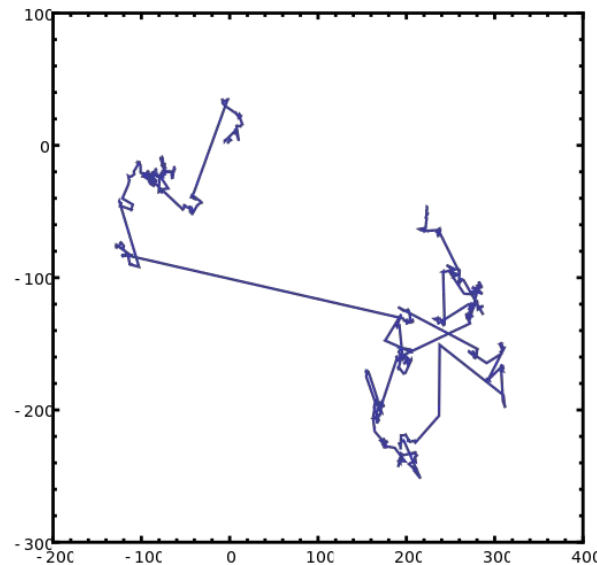
# Tradeoff between Accuracy and Speed

- Breakthroughs improve both accuracy and speed
  - Factorized Convolution (GoogleNet)
  - Skip connection (ResNet)
  - Fully Convolutional Network
  - Better Loss Function
  - Batch Normalization



# Searching for good model

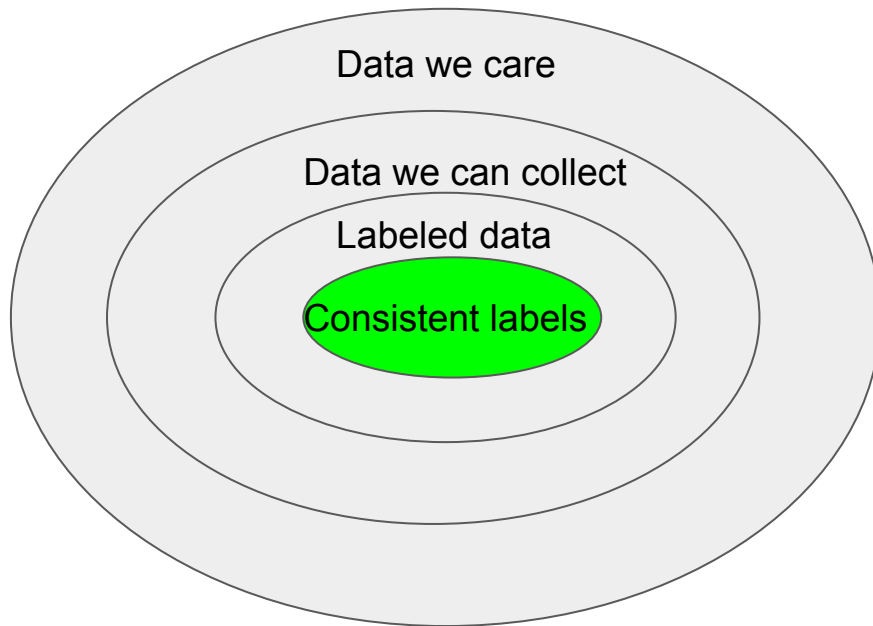
- Repeat: big-step, many baby-steps
  - Big steps helps you explore design space
    - Make wild changes and hope to get better
  - Baby step locally search for local optimum
    - Mostly in the form of control experiments where only one factor changes, to allow for later combination of factors
    - Should be densely logged



Lévy flight, one way animals find food.

# Training with Synthesized Data

# The scarcity of labeled data



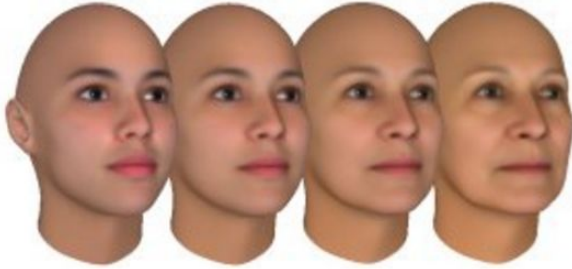
# The scarcity of labeled data

- The data may be sensitive
  - Unlabeled ID card costs **70 yuan/image**.
- Data like videos are hard to label
  - High frame rate x High resolution
- Even hard to define labels
  - Where's the border between face and non-face in this image? =>

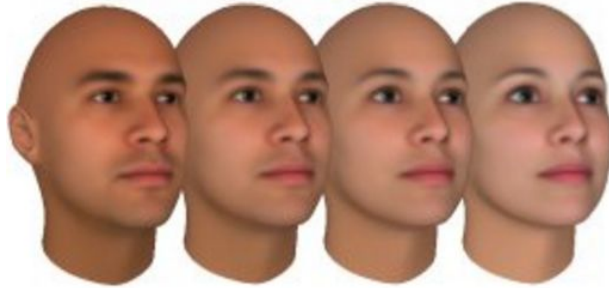




# Computer Graphics can handle 3D



*Age Morphing*

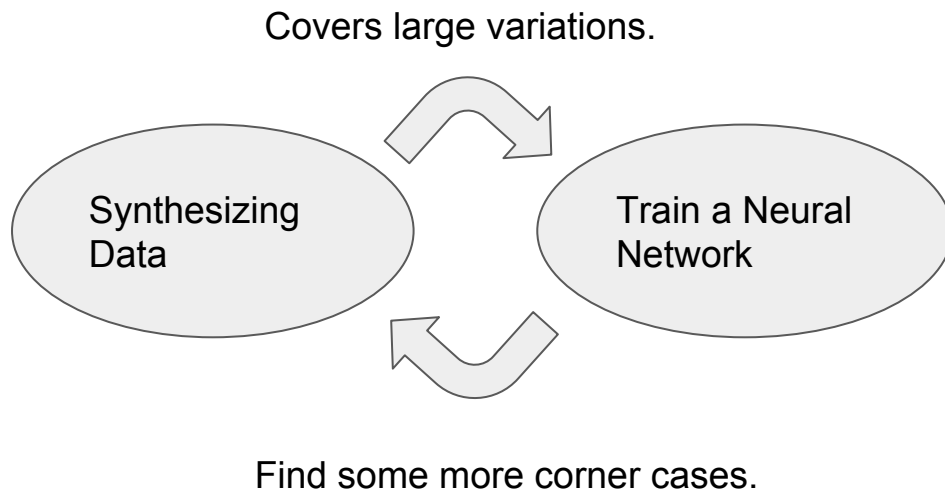


*Gender Morphing*



*Skin Texture*

# Fast evolution without human labeling



# Computer Graphics approach

- Different standards of “real”-looking when used for NN training
- Synthetic data makes up **all** training data for ID cards.
  - after proper augmentation
- Engineering intensive



# Neural Network Synthesizing

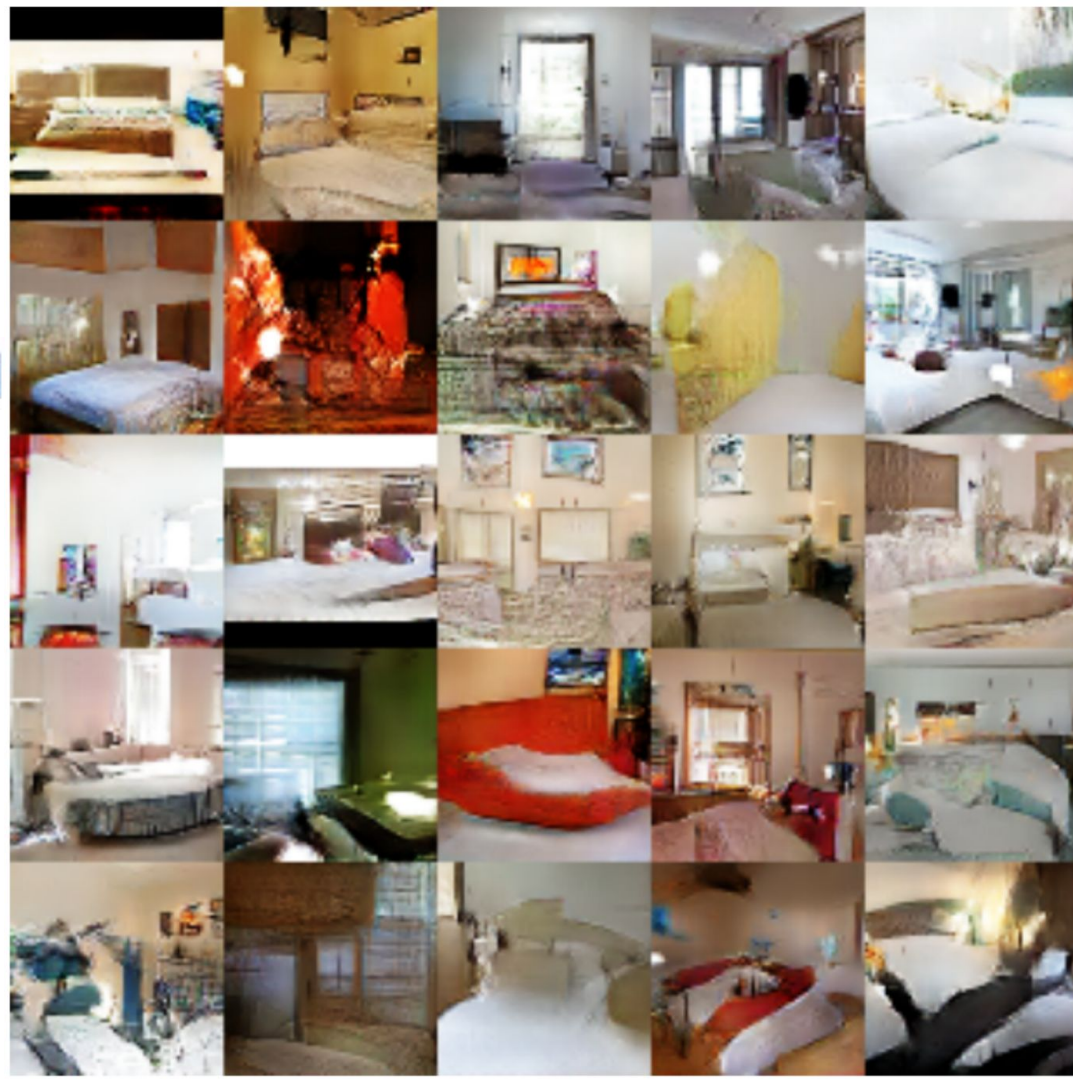
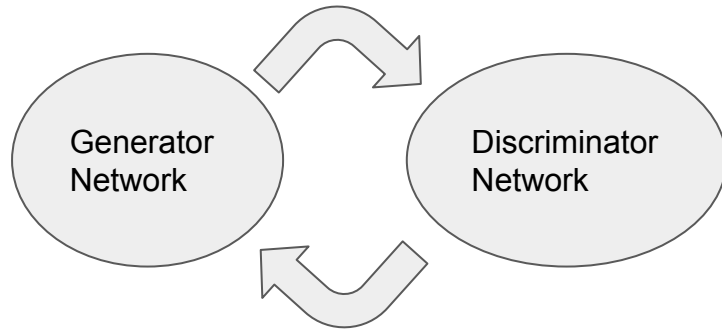
- “Inverse classifiers”
  - CNN: [Synthesizing chairs from attributes](#)
- Generative
  - RNN: [Creating non-existent Chinese char](#)



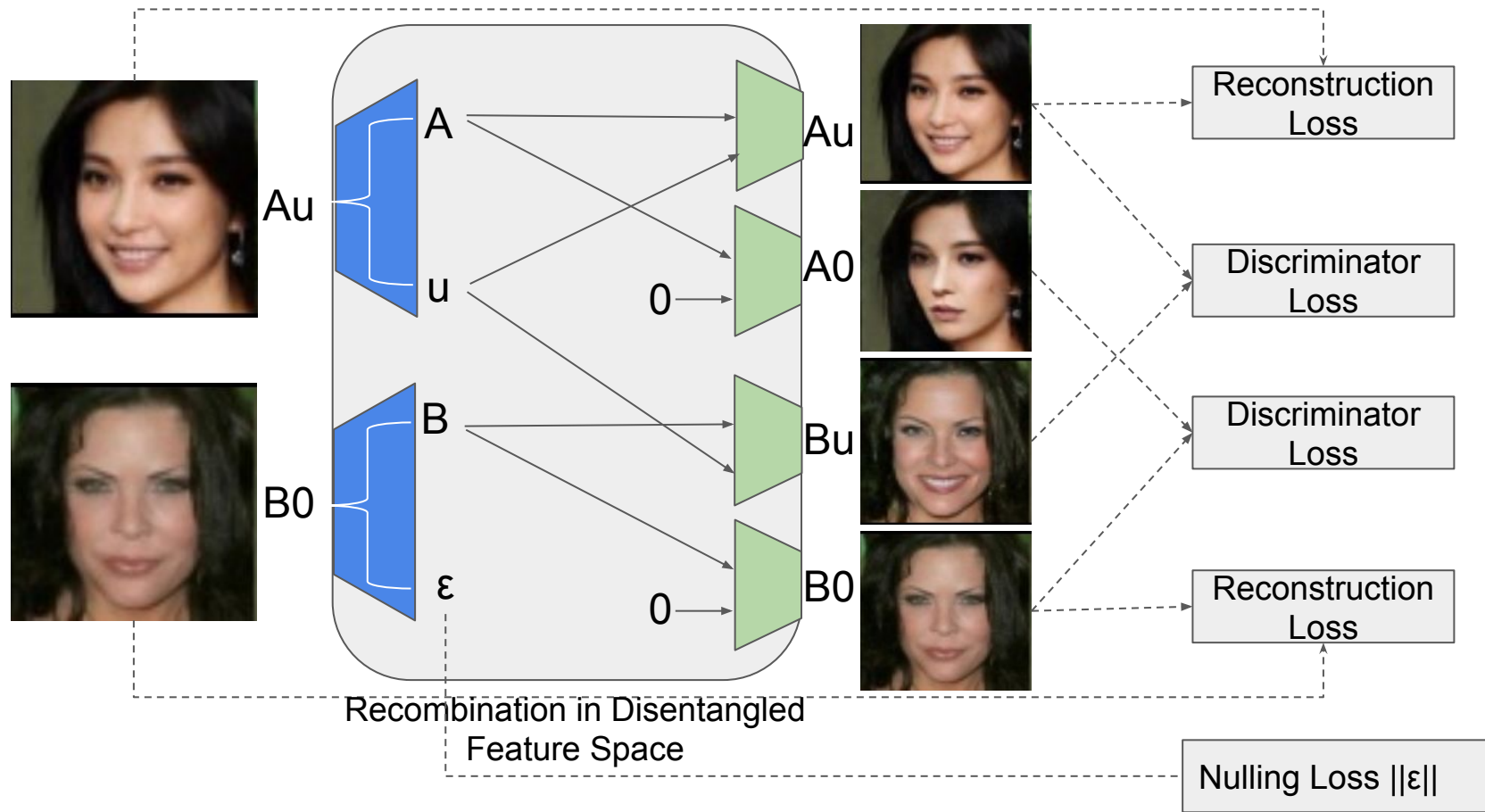
# Generative Adversarial Network

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x})] \\ + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

Take fake  
as true.



# GeneGAN



# Summary

## Solving Problems by Deep Learning

Upside	Downside
Can work on practical problems	Need deal with dirty details
Can context switch when model starts training	Long time to receive feedback
End-to-end pipeline boosts performance	Hard to peep into the all-in-one black box
Many techniques for improving quality	Hyperparameter search space large

# References

- Chapter 11, Deep Learning  
<http://www.deeplearningbook.org/contents/guidelines.html>
- Advice for applying Machine Learning  
<https://see.stanford.edu/materials/aimlcs229/ML-advice.pdf>



Backup after this slide