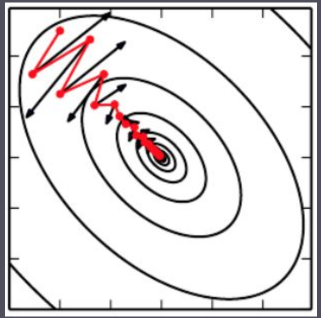
Deep Learning

Part II: Practical considerations

By: *Andreas Damianou*
damianou@amazon.com

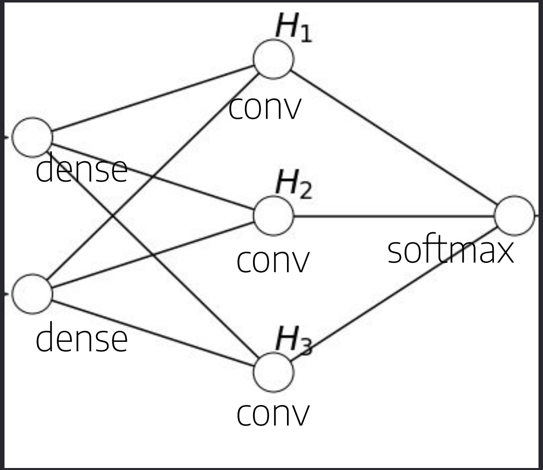


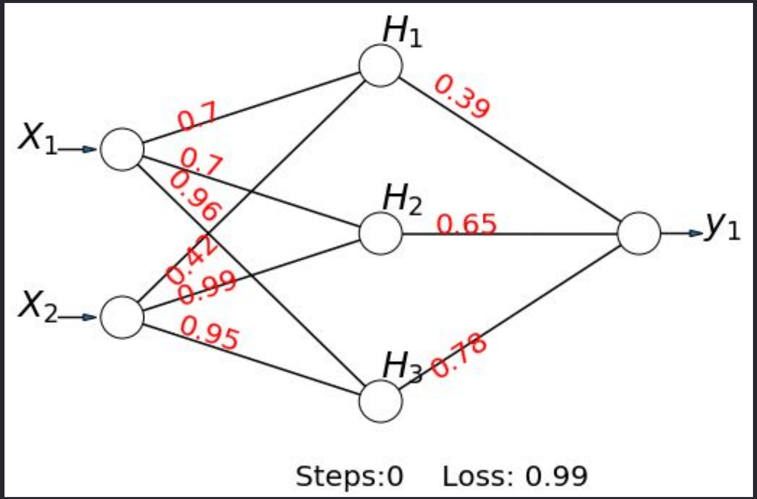
DSA 2020

Motivating example: cassava disease classification

Mwebaze et al. 2019: https://arxiv.org/pdf/1908.02900.pdf

Deep Learning in a nutshell





Create model

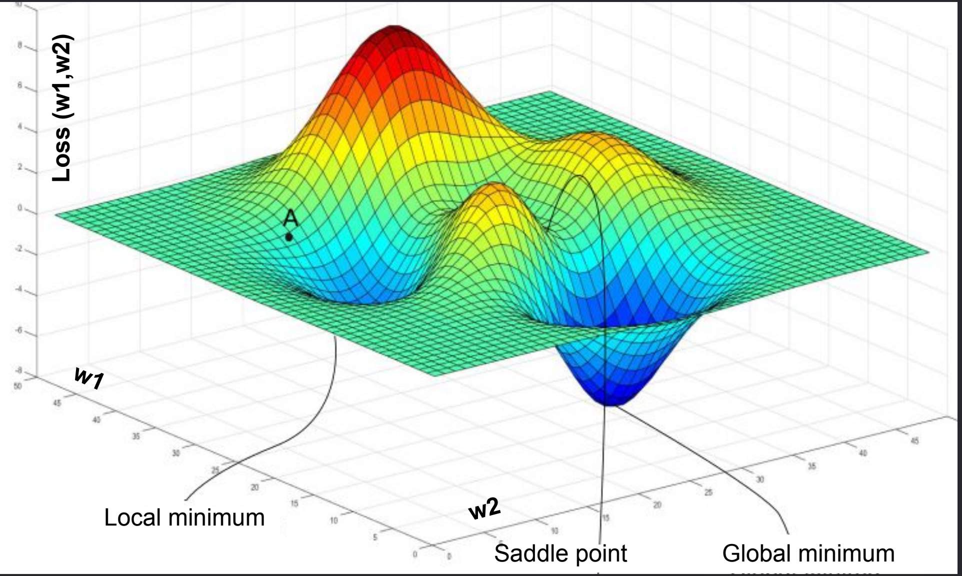
```
m = Input(*img_size)
m = Dense(d)(m)   # contains Weights
m = Conv()   (m)
        ...
```

Optimize model

```
m.fit(data)
```

Loss landscape

- Optimization: find a setting of parameters that minimize the loss.
- Gradients used to **navigate** the loss landscape.
- Non-convexity and high dimensionality cause issues.

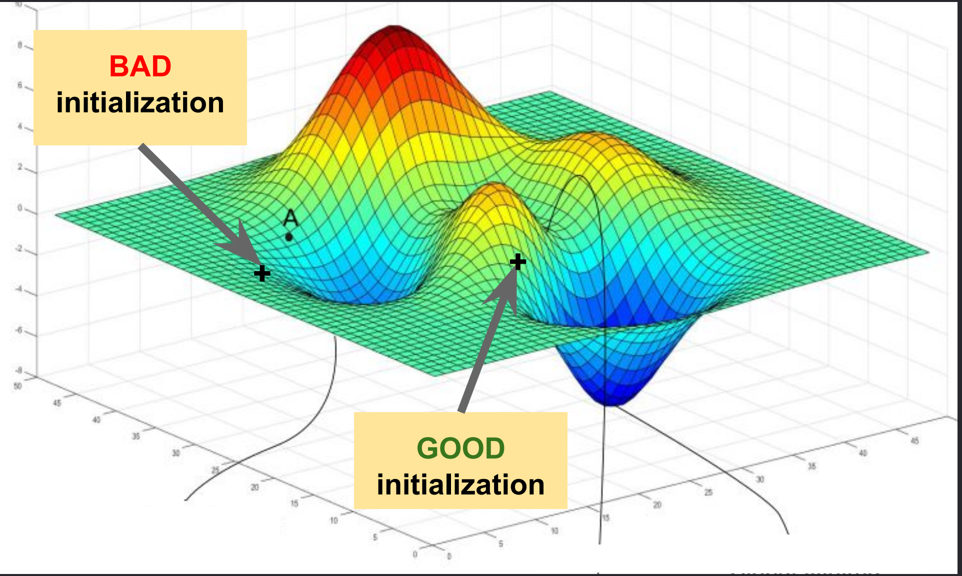


Img credits: https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

Lesson 1: Initialization matters!

Bad initializations:

- Close to local minima
- Areas where gradients vanish/explode



Lesson 1: Initialization matters!

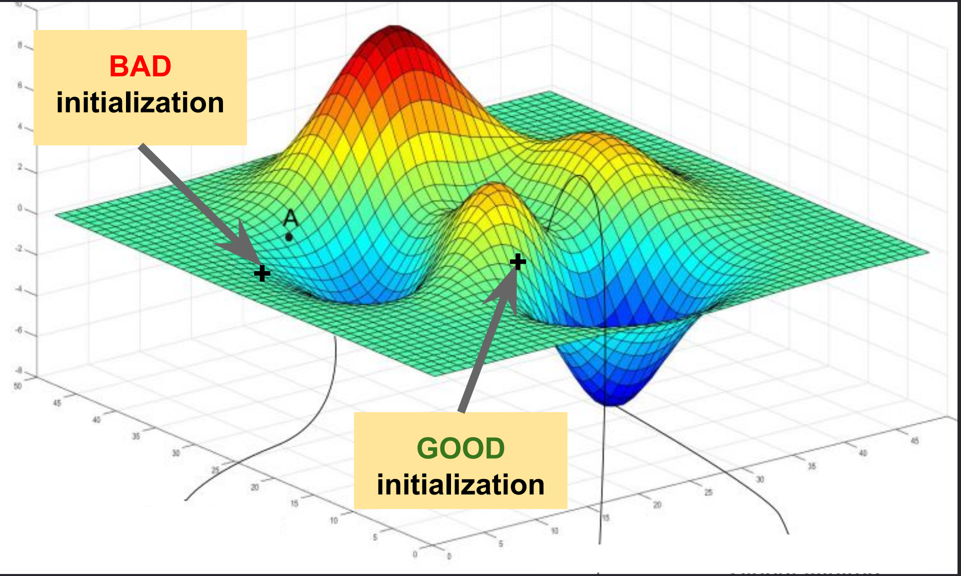
Bad initializations:

- Close to local minima
- Areas where gradients vanish/explode

Rules of thumb*:

- Mean of activations ≈ 0
- Activation variance across layers ≈ same

$$w \sim \mathcal{N}(0, 1/\#\text{neurons\_in})$$

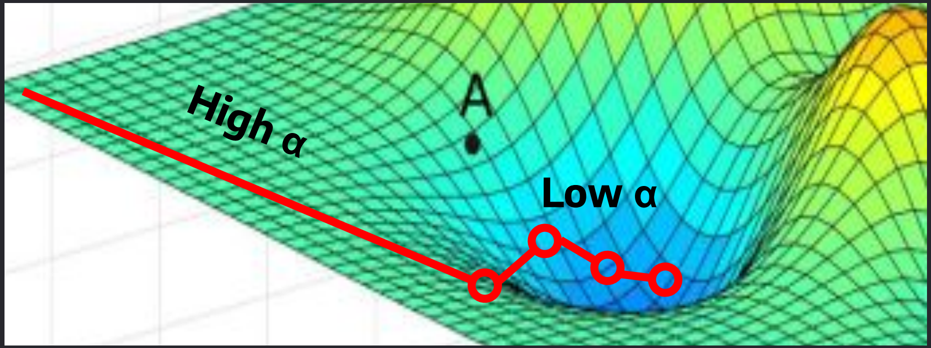


* Glorot & Bengio 2010

Lesson 2: Navigate the loss landscape cleverly

- Adaptive learning rate

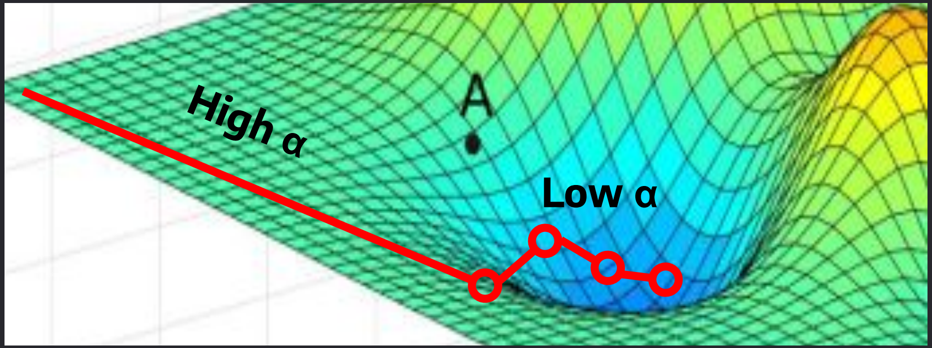
```
W -= α(t) * W_grad
```



Lesson 2: Navigate the loss landscape cleverly

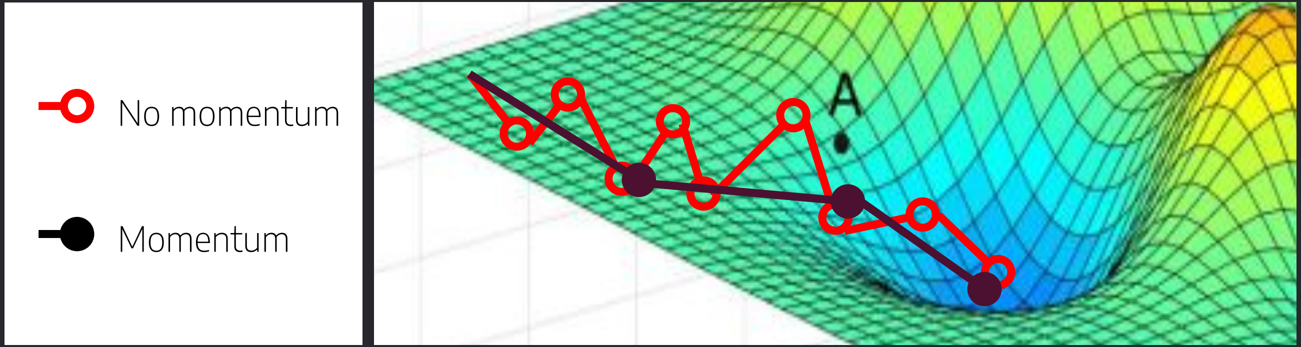
- Adaptive learning rate

```
W -= α(t) * W_grad
```



- Momentum

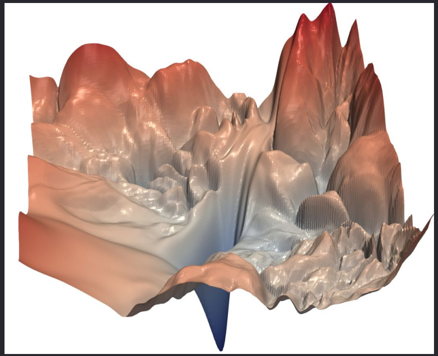
```
W -= α * W_grad
   + b * W_grad_prev
```



Lesson 3: Improve properties of loss landscape

- Batch-normalization

```
m = BatchNormalization()(m)
```



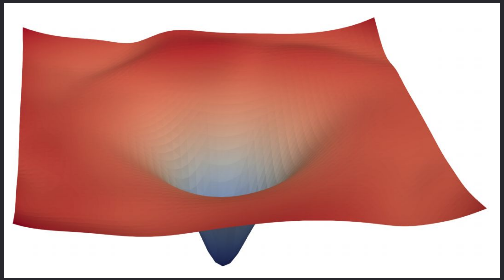
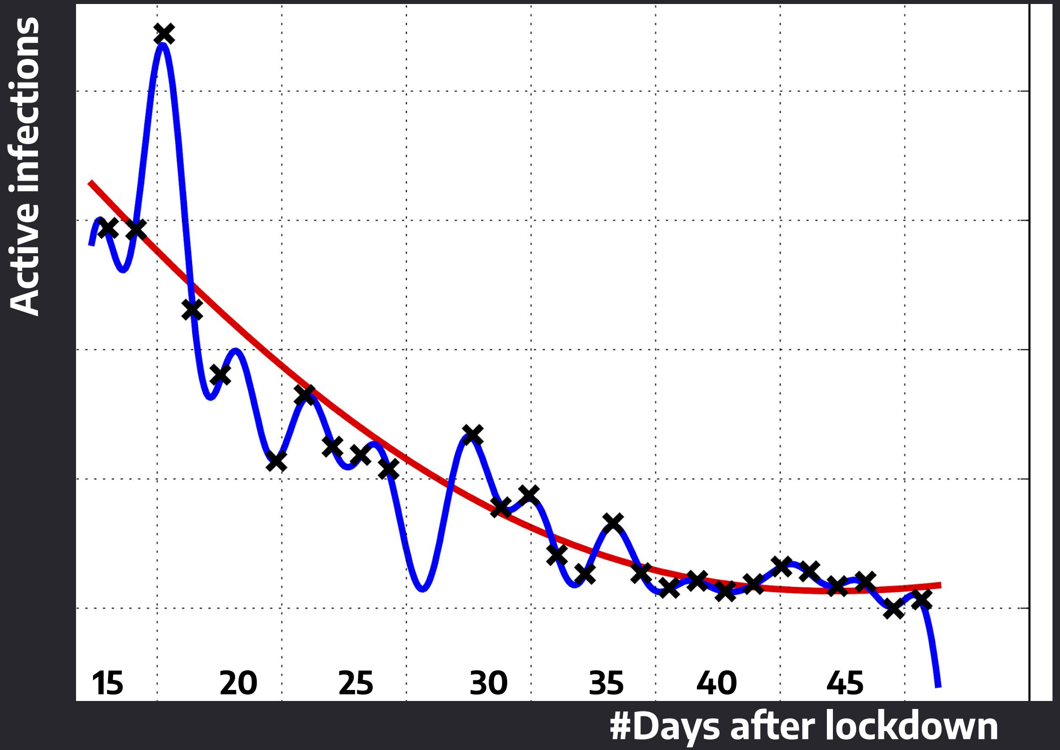


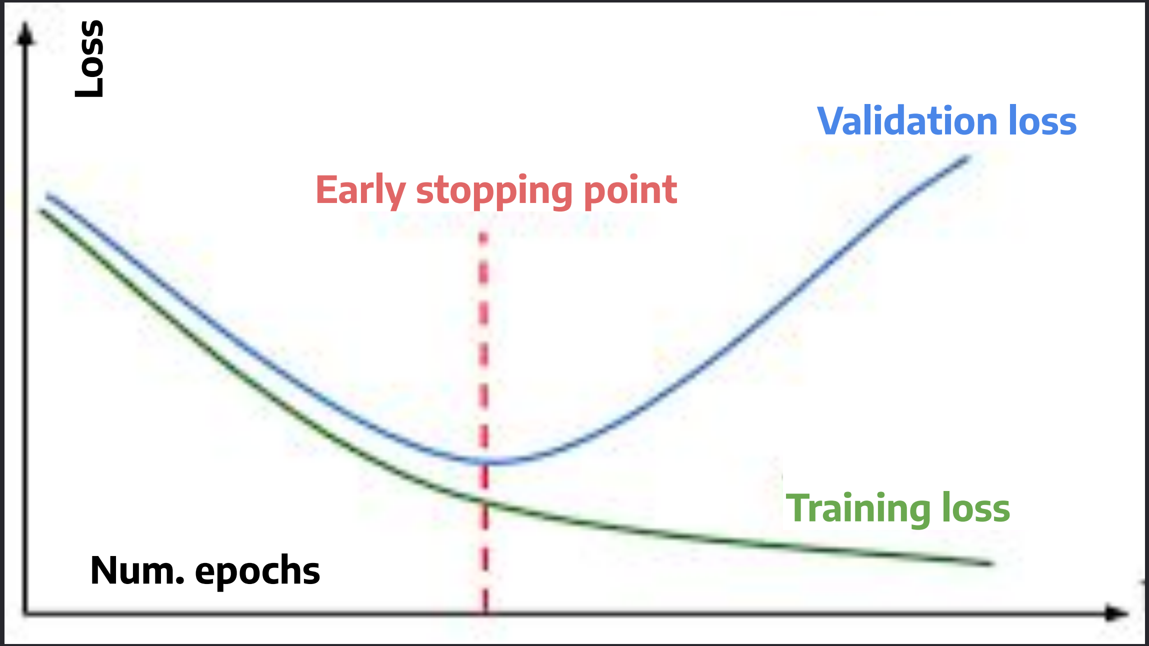
Image:Resnet

Lesson 4: Avoid overfitting



Doesn't mean we have to smooth data;
it means that in the absence of "strong" evidence we shouldn't make "strong" inferences.

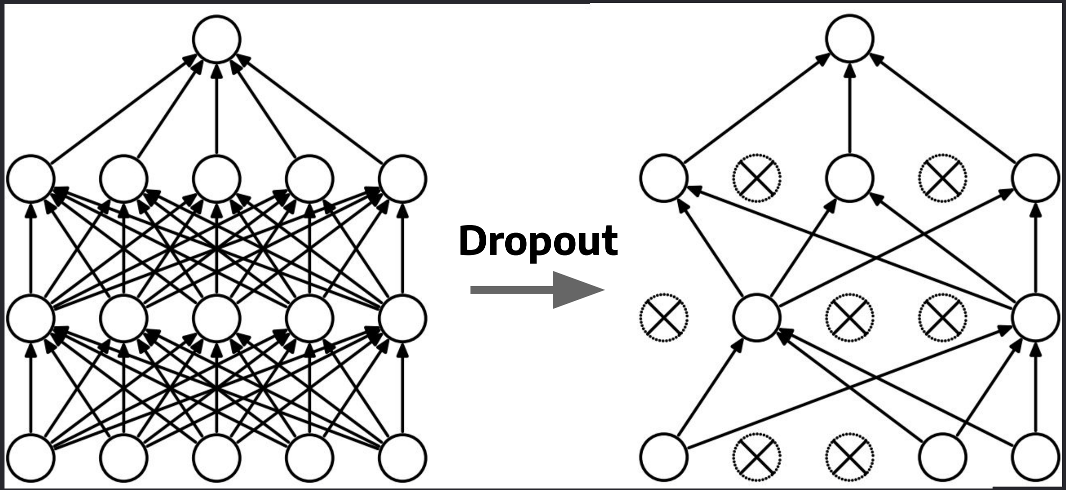
Lesson 4: Avoid overfitting [Early stopping]



Lesson 4: Avoid overfitting [Dropout]

- Randomly drop units during training

- Prevents unit co-adaptation and overfitting

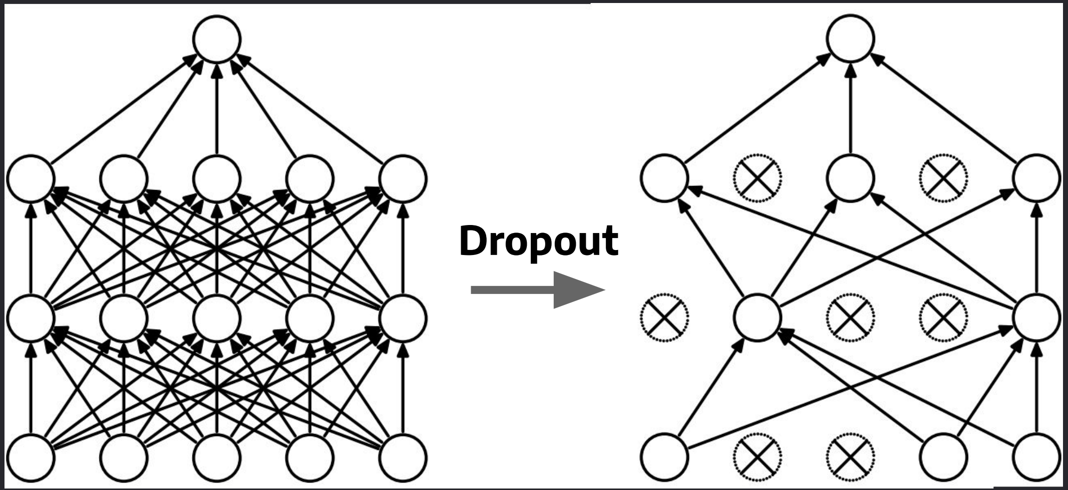


Lesson 4: Avoid overfitting [Dropout]

- Randomly drop units during training

- Prevents unit co-adaptation and overfitting

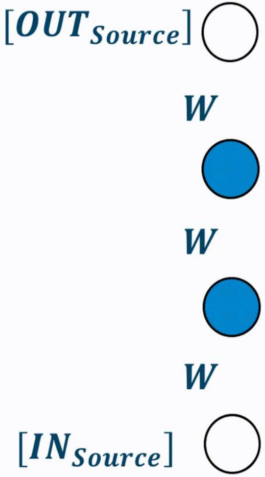
```
m = Input(*img_size)
m = Dense(d)  (m)
m = Dropout(0.5)(m)
      ...
```



Lesson 5: Transfer learning

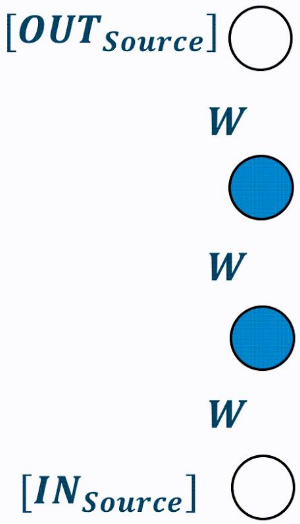
- Solves many of the issues because it starts from a “good” parameter setting that works well for another, related task.

Given



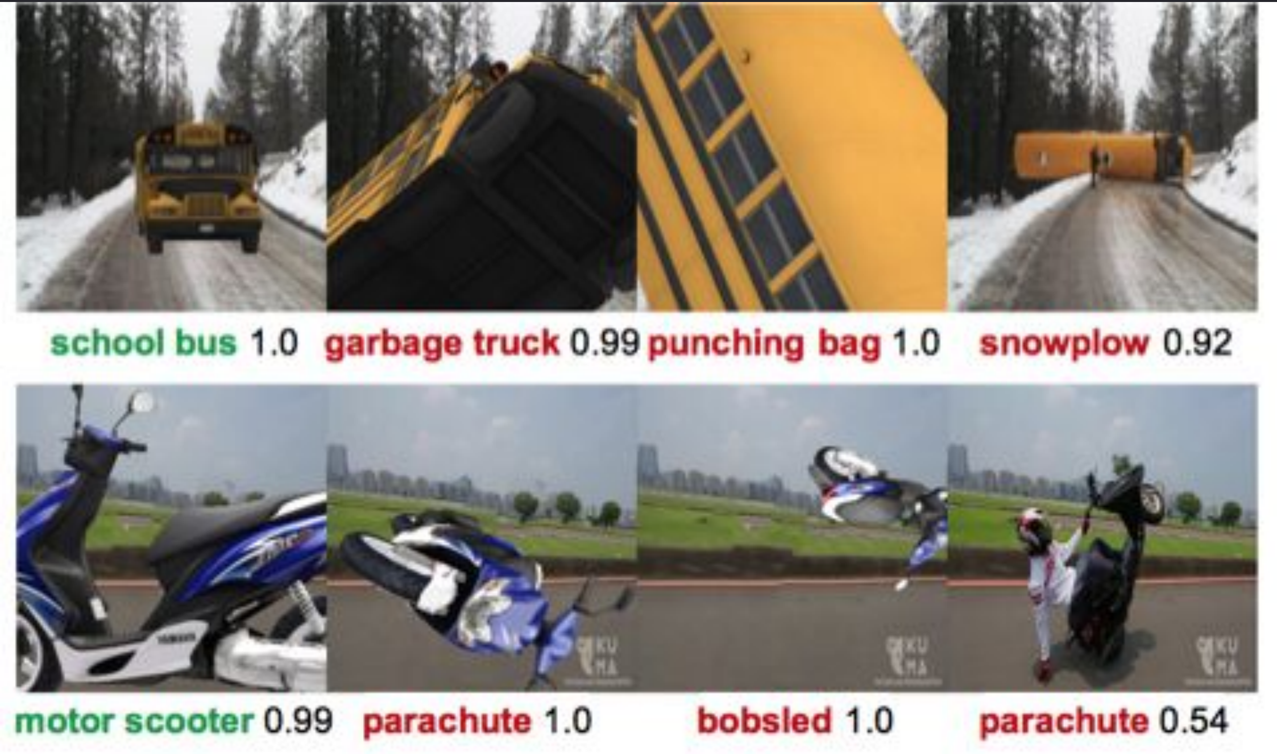
Transfer learning with feature extraction

Given



Failure cases

Failure cases: unfamiliar poses



* Alcorn et al. 2019

Failure cases: data issues

- Real data is messy (e.g. poor focus images)
- Real data is often limited

Failure cases: unexplainable predictions

Quinn et al. 2016: http://proceedings.mlr.press/v56/Quinn16.pdf

Other practical issues regarding learning

- Gradient properties (exploding, vanishing)
- Scalability & Storage (huge networks)
- Numerical issues
- Mismatch between training & test distribution
- Data inefficiency
- Continual learning
- …

Take home messages

- Understanding the loss landscape
- Initialization matters
- Navigate the loss landscape cleverly (adaptive learning rate; momentum)
- Make loss landscape better behaving
- Avoiding overfitting (early stopping; dropout)
- Transfer learning
- Consider failure cases