AI Engineer Training: VI
In the Era of Deep Learning

IT21 Learning
Alvin Jin

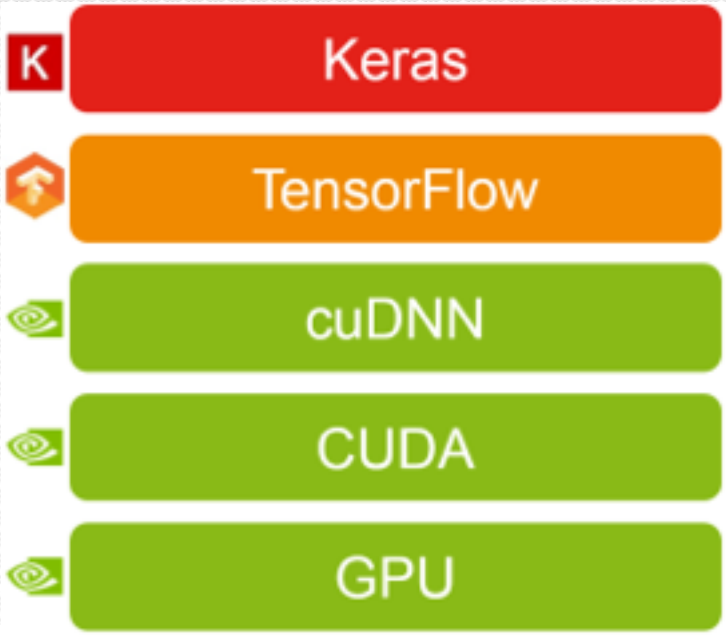
Agenda

- Deep Learning in Computer Visions:
  - Setup Deep Learning on GPU
  - Fight Overfitting
- Case Studies:
  - Kaggle Cats vs. Dogs Classifications

GPU and CUDA

- GPU(Graphics Processing Unit):
  - hundreds of simpler cores
  - thousand of concurrent hardware threads
  - maximize floating-point throughput
- CUDA(Compute Unified Device Architecture)
  - a parallel programming model that enables dramatic increases in computing performance by harnessing GPU
- cuDNN(CUDA Deep Neural Network library)
  - a GPU-accelerated library of primitives for neural networks.
  - it provides highly tuned implementations for: convolution, pooling, normalization, and activation layers.

Deep Learning Software Stack



Case Studies: Cats vs. Dogs

- The challenge was published on Kaggle in 2013, with varying image resolutions.
- The training data contains 25,000 images of dogs and cats
- https://www.kaggle.com/c/dogs-vs-cats/data

SGD, Batch and Epoch

- Stochastic Gradient Descent (SGD), computes the gradient and updates the weight matrix on each training sample.
- SGD makes computation faster, while using whole dataset makes vectorization less efficient.
- Instead of computing gradient over either whole dataset or single sample, we evaluate the gradient on mini-batch (32, 64, 128, 256), then update our weight matrix.
- Epoch is one forward pass and one backward pass of all the training samples.

Overfitting

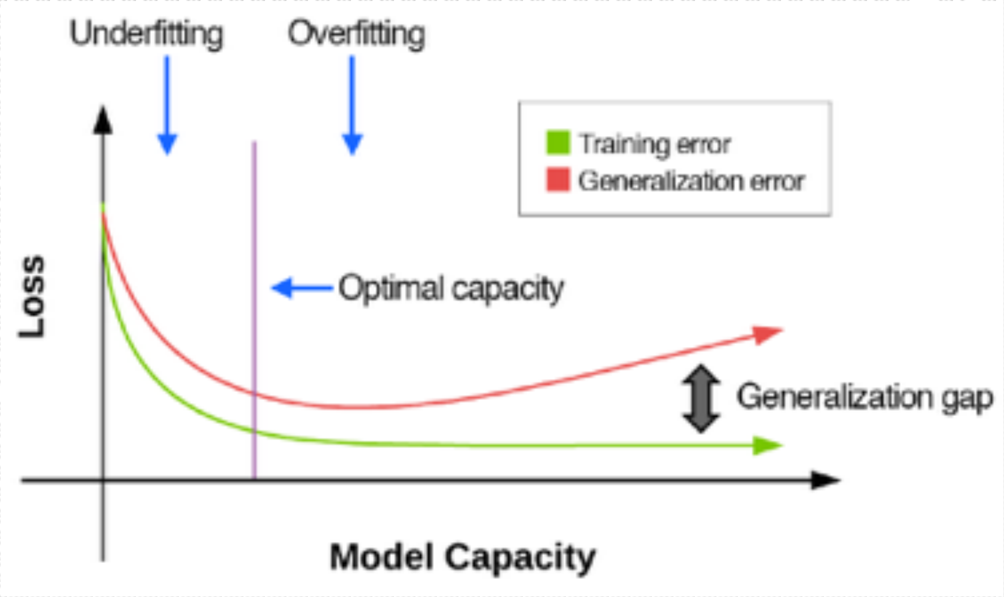
- Overfitting occurs when the gap between training and validation loss is too large.
- Indicates the network is modelling the underlying patterns in the training data too strong, while doesn't work well for never seen validation data.
- As long as the loss gap between training and validation doesn't increase dramatically, there is an acceptable level of overfitting.

Overfitting Solutions

- Add more training data
- Reduce capacity of model
- Applying regularization technologies
- Batch normalization

Loss vs. Model Capacity

- As model capacity increases, training and validation loss/accuracy start to diverge from each other



Regularizations

- Regularizations help us control model capacity, ensuring it has ability to generalize and eliminate overfitting.

- Common regularization approaches:
  - Regularization on Loss
  - Dropout
  - Data augmentation
  - Early stopping

Data Augmentation

- Data augmentation generates new training samples from the original ones without the classes labels changed to increase model generalizability.

- During the training process, we randomly alter the training images by applying random transformations to them:
  translation, rotation, resizing, and shearing, horizontal flips, etc.

Regularization on Loss

- Update the loss function and the weights update rules, add an additional parameter to constrain the capacity of the model

$$W = W - \underbrace{\boxed{\alpha}}_{\text{learning rate}}\underbrace{\boxed{\nabla_W f(W)}}_{\text{gradient}} - \lambda R(W)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda R(W)$$

- Regularization penalty R(W) is a function operates on the weight matrix.

- If *lambda* is very large, the mode will lead to underfitting.

Regularization Penalty

- L1 regularization(Lasso Regression):
  - intends to drive small weights to zero to remove some features for feature selection.

$$R(W) = \sum_i \sum_j |W_{i,j}|$$

- L2 regularization(Ridge Regression):
  - intends to drive large weights to values closer to zero to improve generalization.

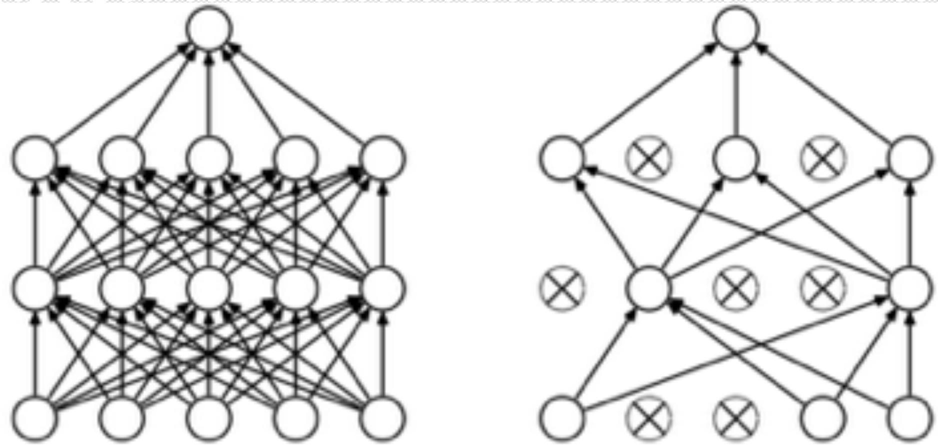
$$R(W) = \sum_i \sum_j W_{i,j}^2$$

L2 Regularization

- The sum of squares in the L2 regularization penalty discourages large weights in our weights matrix.
- Dimensions with larger weight values can almost single handedly control the output prediction of the classifier, and lead to overfitting.
- The final classifier is encouraged to consider all features with small weights, rather than a few strong features.

Dropout Layers

- Dropout layers randomly disconnect inputs from the preceding layer to the next layer in the network.
- After forward/backward pass are computed for this mini-batch, **re-connect** the dropped connections for the next mini-batch.



(a) Standard Neural Net (b) After applying dropout.

Dropout Layers

- Dropout(2014) is to reduce overfitting by explicitly altering the network architecture at the training time
- Randomly dropping ensures that no single unit in the network is responsible for "activating" when presented with a given pattern.
- Instead, multiple units will activate when presented with similar inputs, to train model to generalize.
- It is most common to place dropout layers with p=0.5 in-between FC layers

Batch Normalization

- BN layers(2015) are used to normalize the output(feature maps) of a previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.

- The activations leaving a BN layer will have approximately zero mean and unit variance

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}}$$

Batch Normalization

- Applying BN can help prevent overfitting and obtain higher classification accuracy and lower loss in fewer epochs.

- The drawback of BN is that it can slow down the training time by 2 times due to the computation of per-batch normalization.

- If BN is used, less dropout can be used, without losing too much information.

Weight Initialization

- Constant Initialization
  - All weights are initialized with zero or one.
  - Due to the symmetry of activations, each hidden unit will get exactly the same signal.

- Uniform and Normal Distribution
  - LeCun(PyTorch)
  - Xavier(Keras)
  - He Kaiming(Deeper Network).

CNN Properties

- Compositionality
  - Stack and learn of higher-level features based on lower-level inputs
- Translation Invariance
  - the same object with slightly change of position won’t fire up the unit to recognize that object

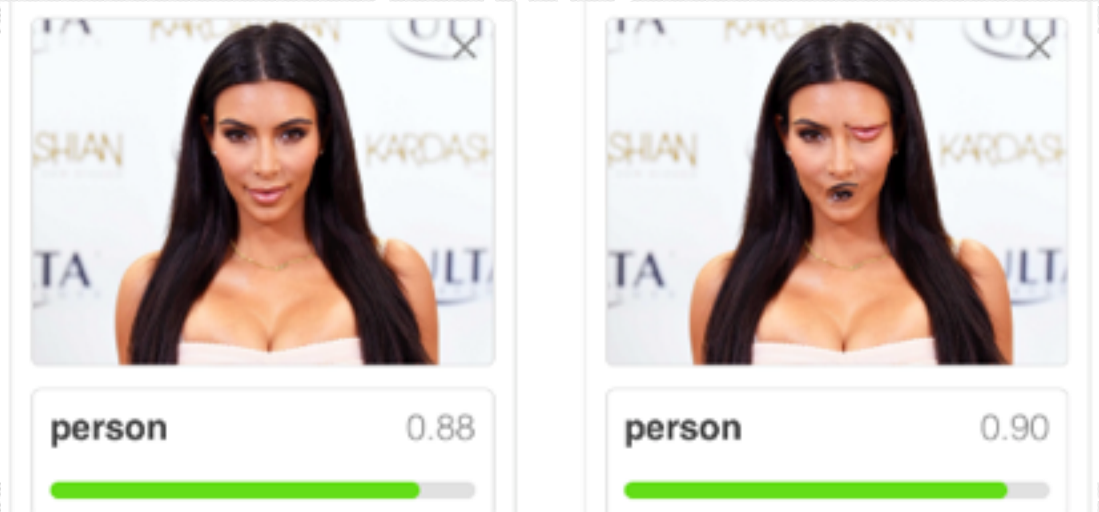
Compositionally

- CNNs can learn spatial hierarchies of patterns. This allows convents to efficiently learn increasingly complex and abstract visual concepts.

- Each filter composes a local patch of lower-level features into a higher-level representation, f (g(h(x))) , to learn more rich features deeper in the network.

Translation Invariance

- After learning a certain pattern in the lower-right corner of a picture, a convent can recognize it anywhere.
- It's unable to identify the position of one object relative to another, it can only identify if they exist in a certain region.



Q & A