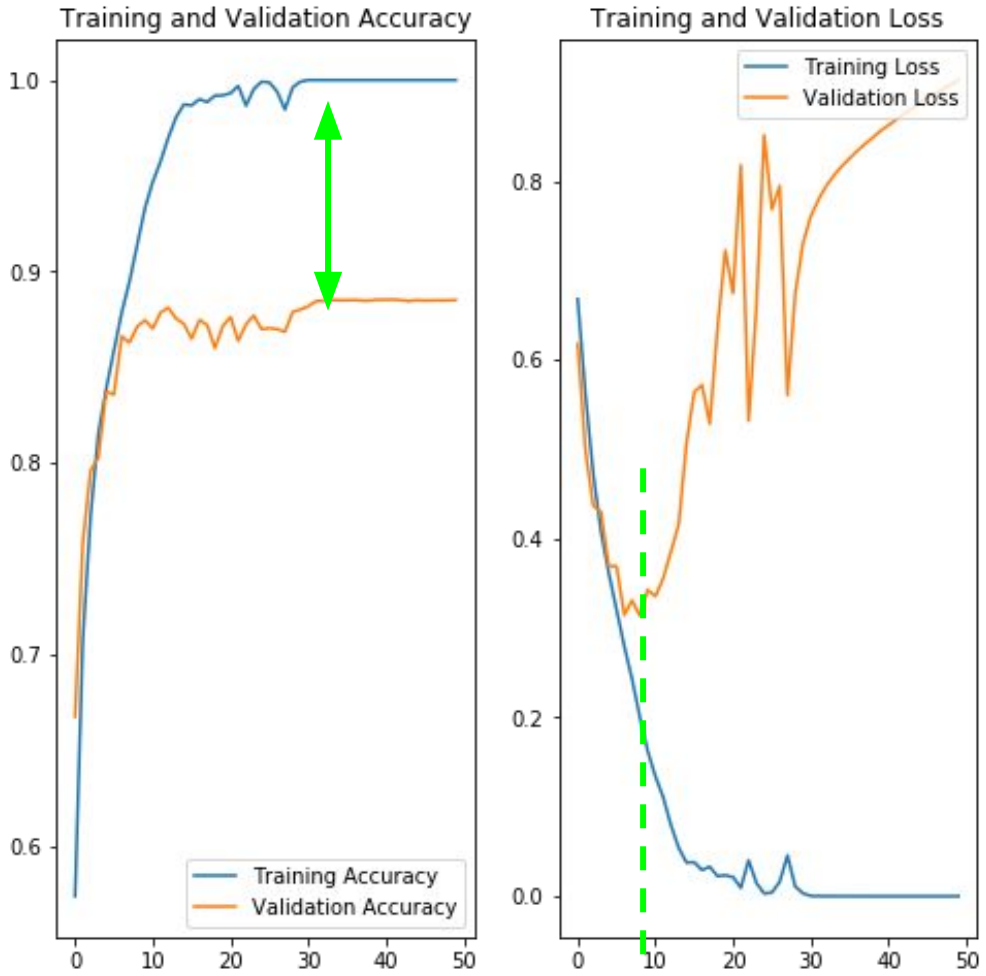
Learning Machine Learning with Kaggle Challenges

(4) Deep Learning (cont'd)

Qiyang Hu
IDRE

Quick Recap

- Dogs-vs-Cats challenges
  - 25,000 training images
  - 15,000 testing images
- Construct our own CNNs
  - 4 Conv layer blocks
  - Flatten layer
  - Dense layer
- Overfitting
  - Memorizing training set too much
  - Missing essence
- How to improve?
  - Need more training data
  - Need regularization



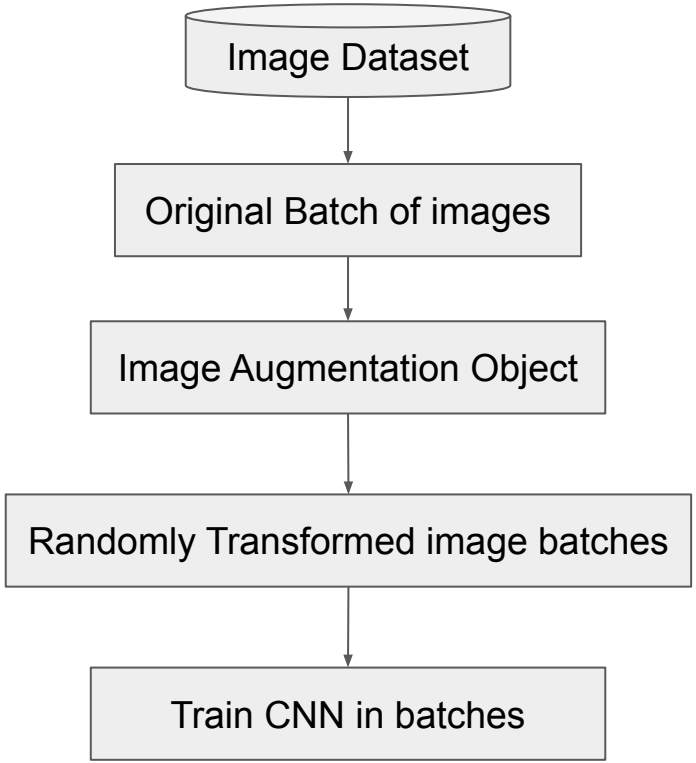
Dataset: the bigger the better, but why?

	VGGNet	DeepVideo	GNMT
Used For	Identifying Image Category	Identifying Video Category	Translation
Input	Image 	Video 	English Text 
Output	1000 Categories	47 Categories	French Text
Parameters	140M	~100M	380M
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words
Dataset	ILSVRC-2012	Sports-1M	WMT'14

Source

How to get more data with “no more”?

- Use data augmentation
  - Various transformations to the available dataset
  - Prevent the irrelevant data
- Types of data augmentation
  - Offline augmentation
    - Performing all the transformations beforehand
    - Good for smaller dataset
  - In-place augmentation
    - Performing transformations in mini-batches
    - Preferred for larger dataset
- Data augmentation in Keras
  - ImageDataGenerator



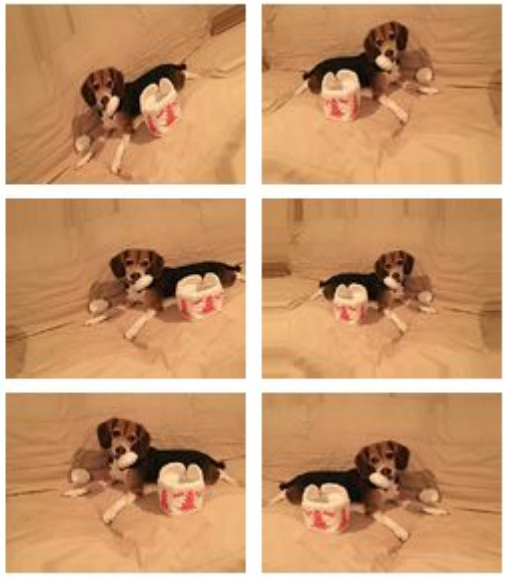
Augmentation Techniques

- Flip
- Rotation
- Zoom & Crop
- Translation

- Gaussian Noise
- Histogram Equalization
- Feature-wise standardization
- ZCA whitening
- Neural Style Transfer (cGANs)

Input Image

Augmented Images



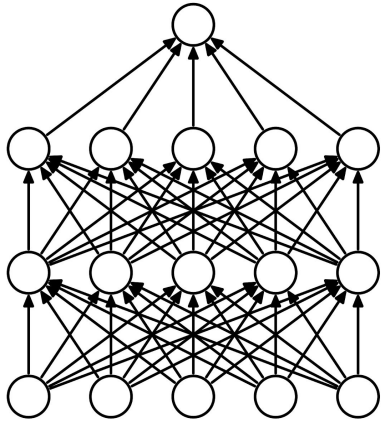
Regularization techniques in deep learning

- Regularizer
  - l1(Lasso), l2(Ridge), l1_l2(ElasticNet) in each layer
    - tf.keras.layer.Dense(..., tf.keras.kernel_regularizer=regularizers.l2(0.01))
  - But not generally used in CNNs
- Early Stopping
  - Keras use Callback function
    - model.compile(...)
      cb = [EarlyStopping(monitor='val_loss', patience=2), ModelCheckpoint(...)]
      model.fit(..., callback=cb, ...)
- Batch Normalization (mutually excludes drop-out, see paper)
  - Define layer without bias
  - Add batch normalization before activation function
- Drop-out

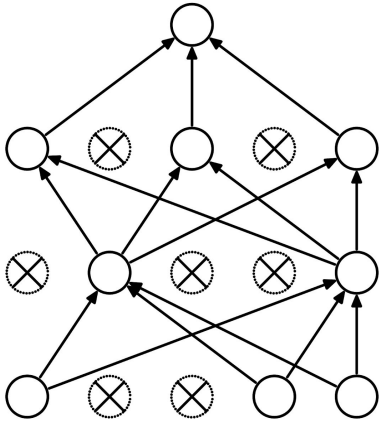
Drop-out technique

- A common problem during DNN training:
  - Imbalanced weights in network:
    some keep very large, others very small
  - Larger weights => well trained
    Smaller weights => not trained that much!

- Dropout: randomly turns off some neurons
  - Forcing networks to train weak neurons
  - Dropout rate: usually 50%
  - Roughly double the iterations to converge
    Training time in epoch is less
  - From Srivastava 2014 paper:
    - 1~2% accuracy boost
      (40% error rate drop for 95% accuracy)



(a) Standard Neural Net



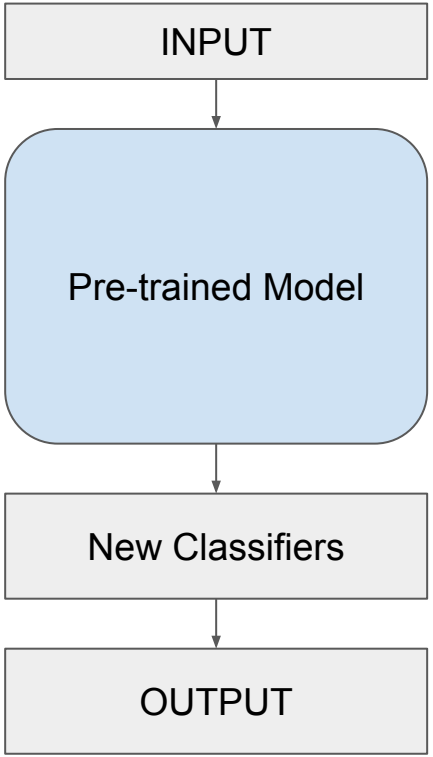
(b) After applying dropout.

Save and Load the model

- Need to save the trained model
  - Colab's active session time is limited.
  - Models can be re-used at user's end (e.g. browser with tf.js or phone with tf.lite)
- Tensorflow provides 3 ways to save/load the model (doc)
  - Through checkpoint callback (`tf.keras.callbacks.ModelCheckpoint`)
    i. Saving weights with tf checkpoints format with a `.ckpt` extension
    ii. Extra work is needed to continue training
  - Serialize the model in HDF5 format
    i. Saving the entire model into a `.h5` file
    ii. Not saving Tensorflow optimizers *yet*, need to re-compile
  - Serialize the model via `SavedModel` method
    i. Saving the entire model into assets folder, `saved_model.pb`, and variables folder
    ii. Keras's `load_model` is compatible with tf serving

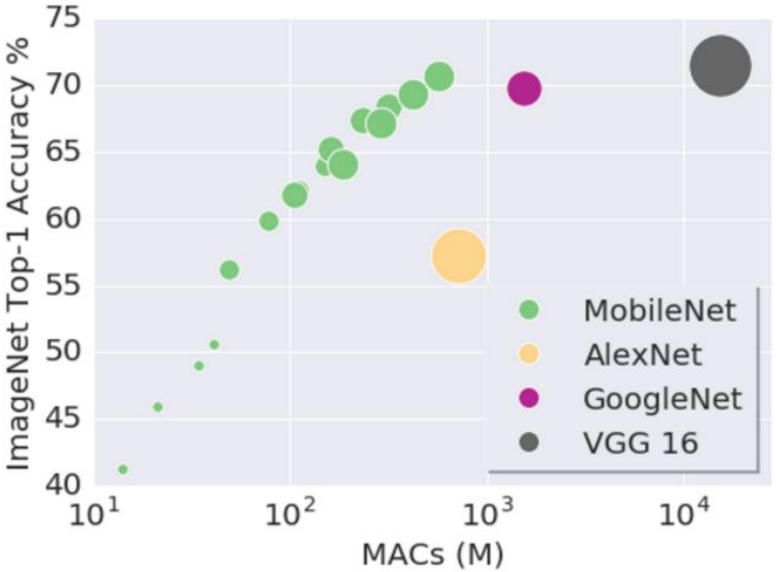
Transfer Learning

- Reusing the developed neural networks
  - Greatly speed up our training
  - Make it mobile
- Image classification
  - Advanced models from ImageNet competition
- Simple steps
  - Match the input size of images from the pre-trained model.
  - Define our new classifiers
    - ImageNet classes: 1280
    - Our classes: 2
  - Freeze the pre-trained layer



MobileNet v2

- Very efficient CNNs (paper)
  - Especially good for mobile vision apps
- From Tensorflow Hub
  - Inception v3 is another choice
- Simple steps:
  - Get the URL from tfhub.dev
  - Define the mobilenet as `hub.KerasLayer`
  - Set the mobilenet layer is not trainable
  - Stack the mobilenet layer and a dense layer as our binary classifier
- For save/reload the model:
  - Need the '`custom_objects`' parameter

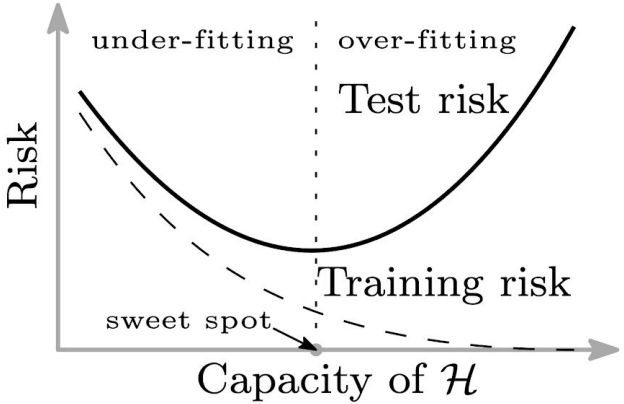


(MACs = Multiply-Accumulates)

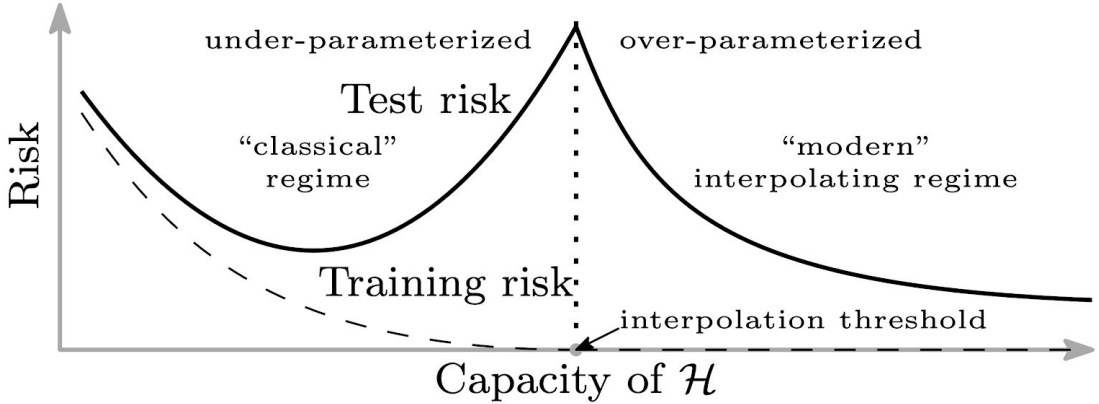
Figure from paper

Are Deep Neural Networks easily overfitted?

- Are Deep Neural Networks easily overfitted?
  - A Blog post
  - From Belkin's 2018 paper:



Classical ML U-shape bias-variance curve



Deep NN Double-U-shape risk curve

Additional Learning Resources

- Within UCLA:
  - I will run similar IDRE workshop series every quarter (IDRE events)
  - 3-day Workshop from QCB: Machine learning with Python
  - M146 – Introduction to Machine Learning (Winter 2020)
  - CS161 – Fundamentals of Artificial Intelligence (Winter 2020)
  - CS260 – Machine Learning (Spring 2020)
- Free online books for solid theories
  - Pattern Recognition and Machine Learning by Christopher Bishop
  - Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville
  - The Elements of Statistical Learning by Jerome Friedman, Robert Tibshirani and Trevor Hastie

Don’t forget to

- Sign in your info to the class
  - To get the email notifications
- Contact me for questions & discussions
  - huqy@idre.ucla.edu
  - Office: Math Sci #3330
  - Phone: 310-825-2011

- Fill out the survey for comments:
  - https://forms.gle/t3f8CztFQpeFFksy6