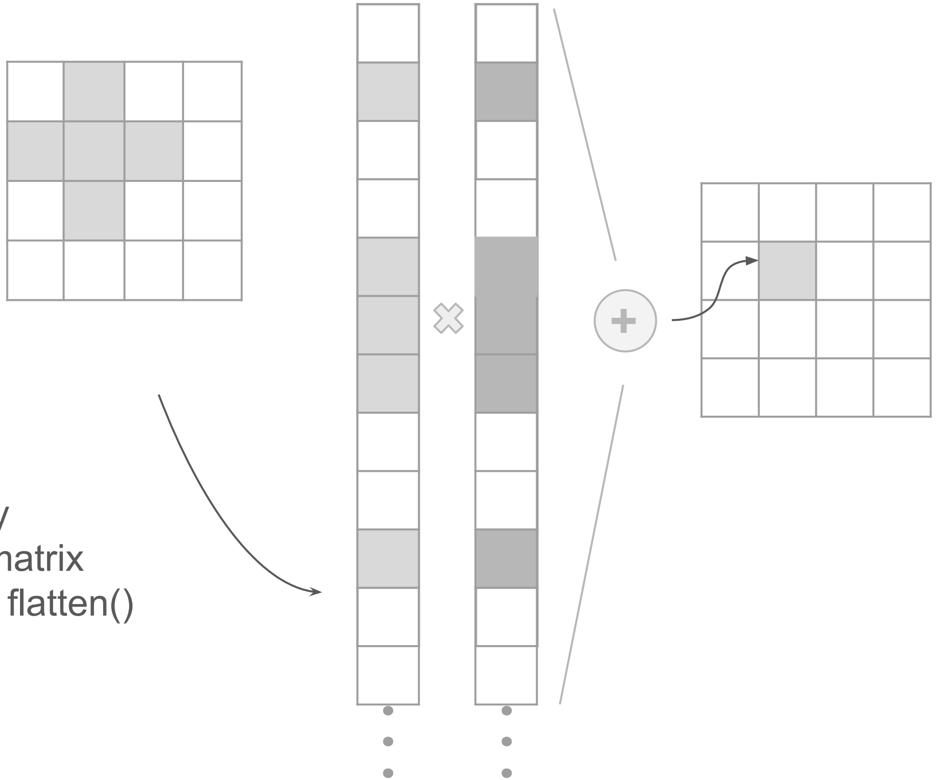
Learning Convolutional Neural Networks (2)

Qiyang Hu
UCLA Office of Advanced Research Computing
Feb 24th, 2023

Follow-up discussions from the last talk

- Is the first column a 1D representation of the original 2D array?

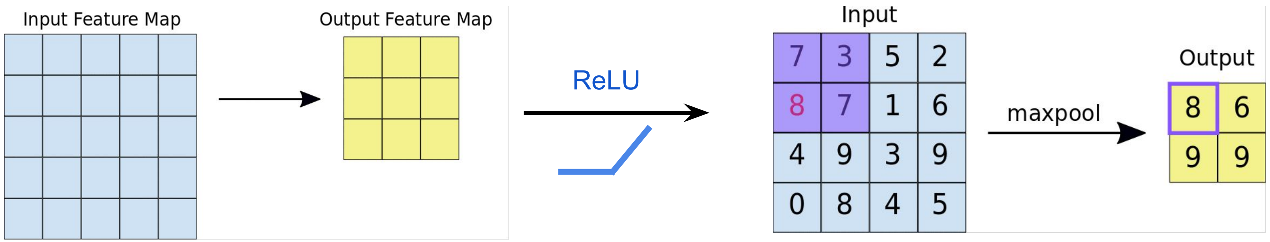


  - ➢ By any customized way
  - ➢ Extra work for sparse matrix
  - ➢ By reshape(), view() or flatten()

Follow-up discussions from the last talk

- Why are there 2 downsampling steps (Conv + Pooling)?

Convolutions to extract as tiles Activation Poolings to downsample



Cases of max pooling might not be appropriate:

➢ The spatial resolution of the feature maps is already low
➢ In semantic segmentation or object detection tasks
➢ Many popular CNNs do not use pooling layers or use them sparingly.

In this talk

bit.ly/**LDL_repo**

Improving our model

01

- Data Augmentation
- Dropout & batch norm
- Demo

Transfer learning

02

- Transferring knowledge
- MobileNet
- Demo

Latest Developments

03

- ViT & Swin-T
- ConvViT & ConvNeXt

In this talk

Improving our model

- Data Augmentation
- Dropout & batch norm
- Demo

01

Transfer learning

- Transferring knowledge
- MobileNet
- Demo

02

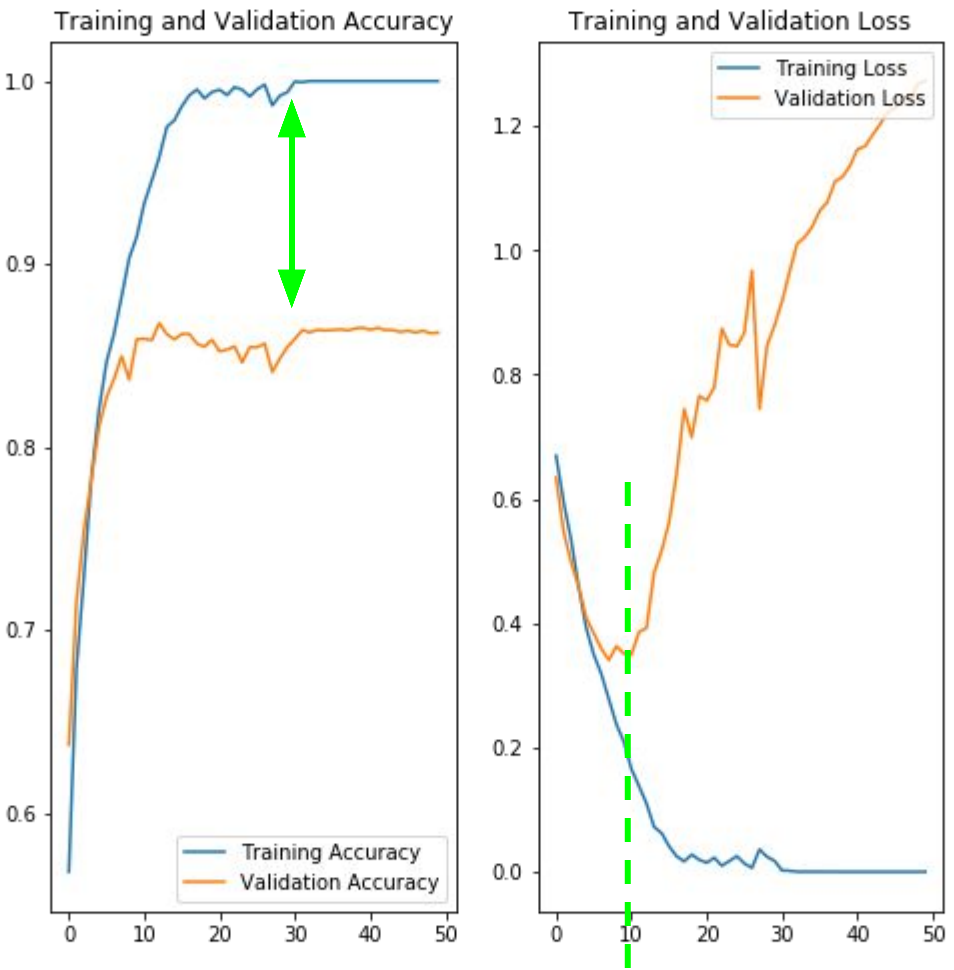
Latest Developments

- ViT & Swin-T
- ConvViT & ConvNeXt

03

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 the essence knowledge
- How to improve?
  - Need more training data
  - Need regularization

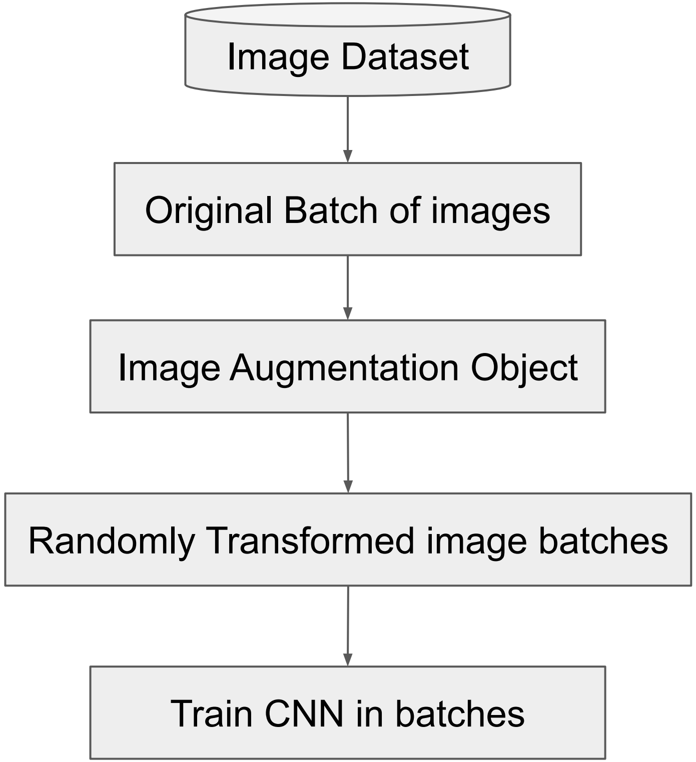


Data and models: the bigger the better, *really*?

	VGGNet	DeepVideo	GNMT	GPT-3
Used For	Identifying Image Category	Identifying Video Category	Translation	Text Generation
Input	Image	Video	English Text	Text
Output	1000 Categories	47 Categories	French Text	Text
Parameters	140M	~100M	380M	175B
Data Size	1.2M Images with assigned Category	1.1M Videos with assigned Category	6M Sentence Pairs, 340M Words	2TB of Internet Text ~ 499 Billion Tokens

From big size to smart size

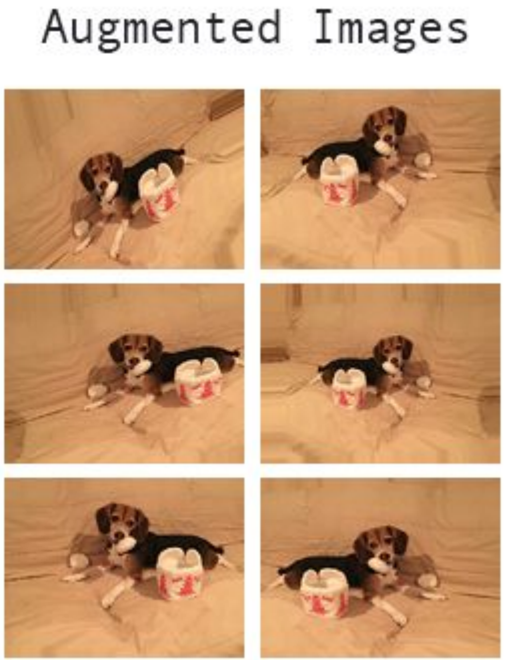
- Using data augmentation
  - To get more data with “no more”
  - Various transformations to the available dataset
  - Prevent the irrelevant data
- Types of data augmentation
  - Offline augmentation
    - Performing all the transformations beforehand
    - Good for smaller dataset
  - In-place augmentation
    - Performing transformations in mini-batches
    - Preferred for larger dataset
- Data augmentation in PyTorch
  - torchvision.transforms



Qiyang Hu

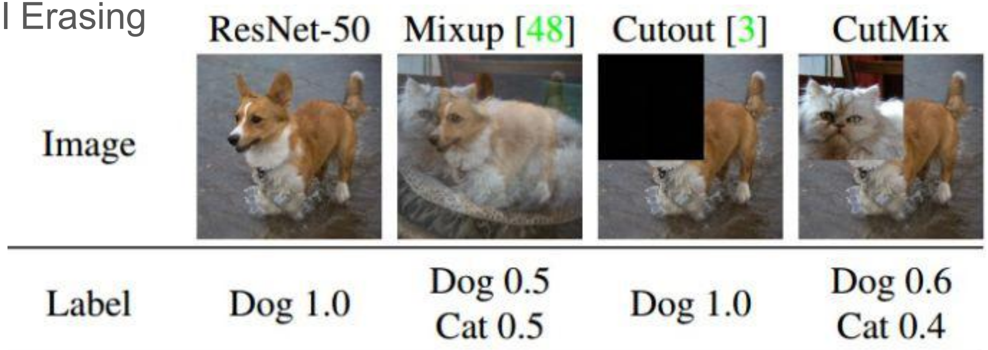
Augmentation Techniques

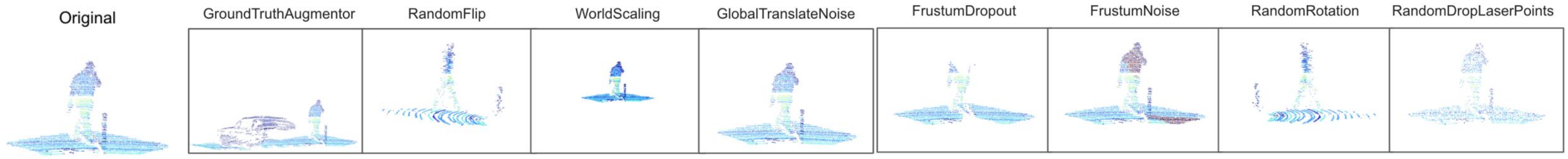
- Flip
- Affine Transformation
  - Rotation
  - Zoom & Crop
  - Translation
- Gaussian Noise
- ZCA whitening
- Histogram Equalization
- Feature-wise standardization
- Neural Style Transfer ([real-time web editing](#))



More Data Augmentation Techniques in CV tasks

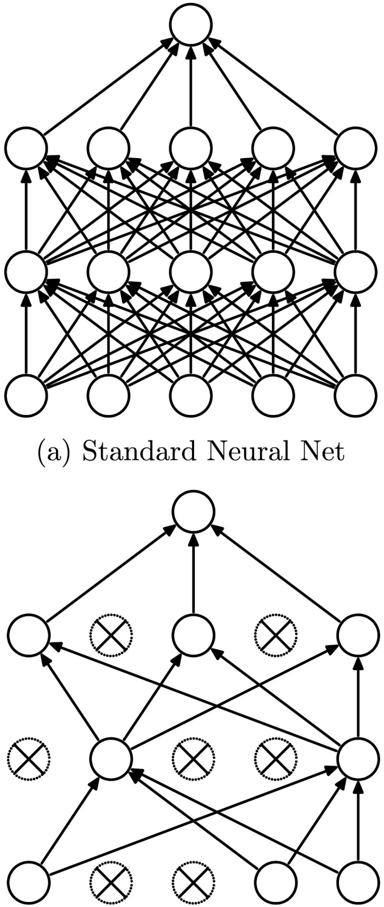
- Random erasing: torchvision.transforms.RandomErasing(p=0.5, scale=(0.02, 0.33), ...)
  - Cutout (masking out random sections): no label change
  - Hide-and-seek, GridMask
  - Object Region Mining with Adversarial Erasing
- Mixup: soft overlapping
- Cutmix/Mosaic: hard masking
  - Cutmix: 2 images mixed
  - Mosaic: 4 images mixed
- 3-D augmentation





Drop-out technique

- Gradient vanishing during DNN training:
    - Imbalanced weights in network:
        - Larger weights => well trained
        - Smaller weights => not trained that much!
- Dropout: randomly turns off some neurons
    - Forcing networks to train weak neurons
    - Dropout rate: default 50%
        - Roughly double the iterations to converge
          Training time in epoch is less
    - Srivastava 2014 [paper]
    - Variations: spatial dropout, etc.
- PyTorch: [torch.nn.Dropout2d()]
    - Implemented by “Inverted dropout” technique
    - Apply to the corresponding layer(s)

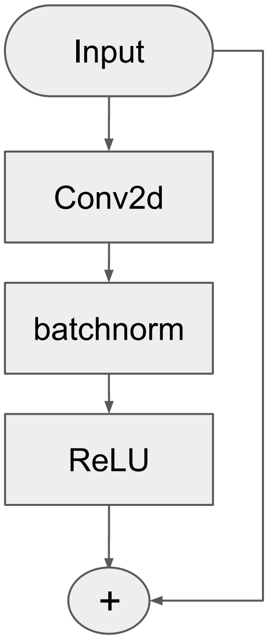


(a) Standard Neural Net

(b) After applying dropout.

Other training techniques in deep learning

- Regularizer
  - l1(Lasso), l2(Ridge), l1_l2(ElasticNet) in each layer
    - weight_decay flag for l2 in pytorch optimizers
- Early Stopping
  - Stop training when validation loss reach minimum
- Batch Normalization
  - Normalize the data (input features) across batches in each mini-batch
  - Add batch normalization before activation function
    - torch.nn.BatchNorm2D(num_features=n_chans1)
- Skip connections
  - Simple trick to add the input (conv1) to the output of a block of layers (conv3)
  - Residual networks (K. He, 2015)
    - Opened the door to hundreds-layer-depth networks (Highway Net, U-Net, Dense-Nets, …)

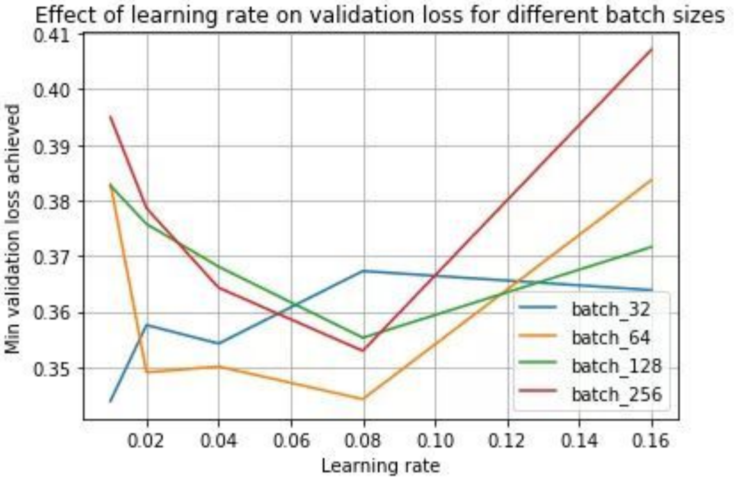


Colab Hands-on

bit.ly/**LDL_cnn2**

Empirical Tips of CNNs

- Design
  - Kernels
    - Shape: square for visual tasks, rectangles for NLPs
    - Size: odd x odd, (3x3, 5x5, 7x7)
  - Number of Conv Layers:
    - Start with a few
    - < 100
- Hyperparameters:
  - Batch size: neither too big, nor too small
    - start from 16, then 8 or 32
  - Learning rate: start from 0.01
    - minibatch size * k ⇒ learning rate * k
  - Use batch normalization
  - Use random search
  - Stride length: DiffStride: learning it through backprop



Source

In this talk

Improving our model

- Data Augmentation
- Dropout & batch norm
- Demo

01

Transfer learning

- Transferring knowledge
- MobileNet
- Demo

02

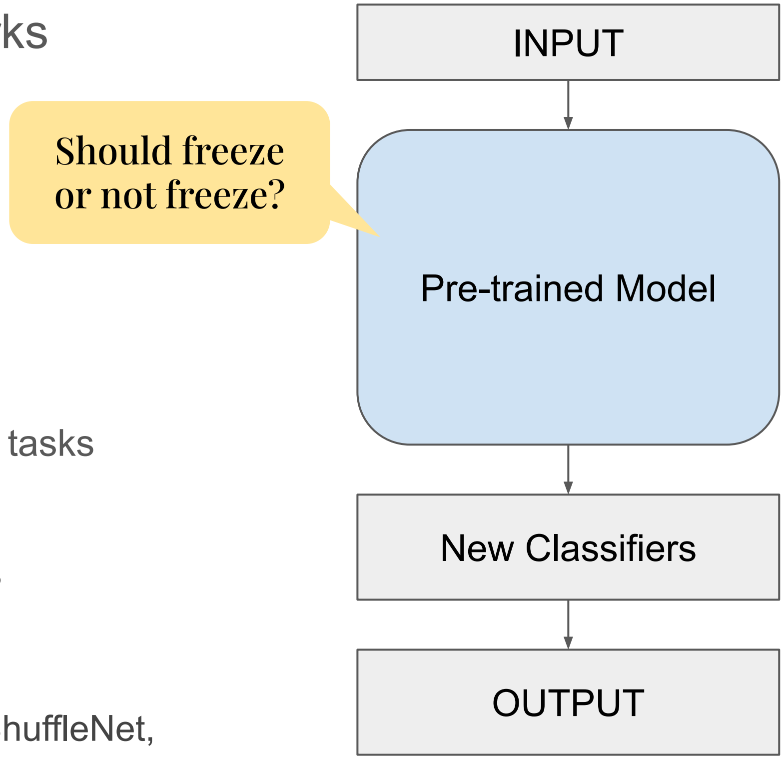
Latest Developments

- ViT & Swin-T
- ConvViT & ConvNeXt

03

Transfer Learning

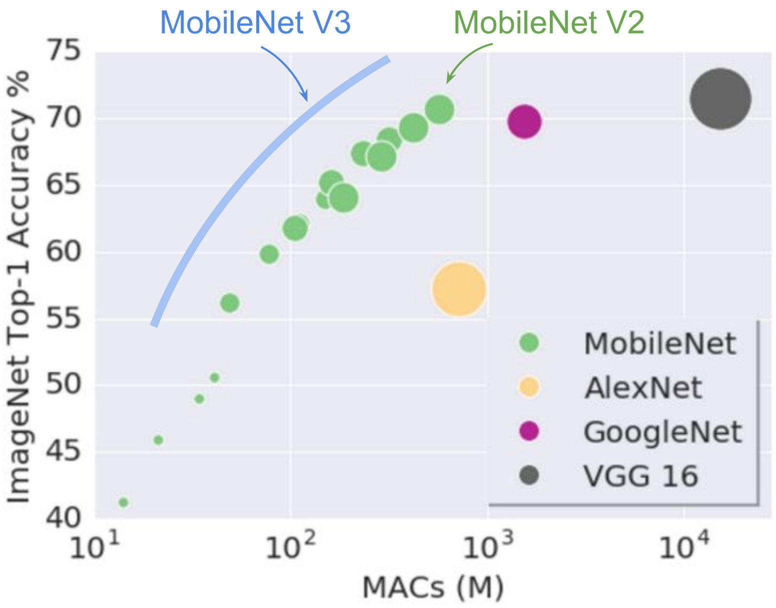
- Reusing the developed neural networks
  - Greatly speed up our training
  - Make it mobile
- Reused model ⇒ feature extractor
  - Pre-trained on a popular generic dataset
  - Transfer the knowledge
    - Match the input
    - Add new layers for specific data and tasks
- Two trends of pre-trained models
  - Gigantic backbones: ResNet, BERT, GPTs
  - Light-fast-efficient backbones:
    - SqueezeNet, MobileNet v1, v2
    - MobileNet v3, TinyYOLO, MnasNet, ShuffleNet, CondenseNet, ESPNet, DiCENet, MixNet, EfficientNet



MobileNet

- Very efficient CNNs (v2 paper & v3 paper)
  - Depthwise separable convolutions
  - Inverted residual with linear bottleneck
  - Squeeze-and-excitation (SE) modules
- Loading the model:
  - MobileNet v2:
    - torchvision.models
    - PyTorch Hub
  - MobileNet v3:
    - large, small and quantized
    - torchvision.models.mobilenet_v3_…
- Modify the classifier layer

model.classifier[3] = torch.nn.Linear(1280, 2)



(MACs = Multiply-Accumulates)

Figure from v2 paper and v3 paper

Colab Hands-on

bit.ly/**LDL_cnn2**

In this talk

Improving our model

- Data Augmentation
- Dropout & batch norm
- Demo

01

Transfer learning

- Transferring knowledge
- MobileNet
- Demo

02

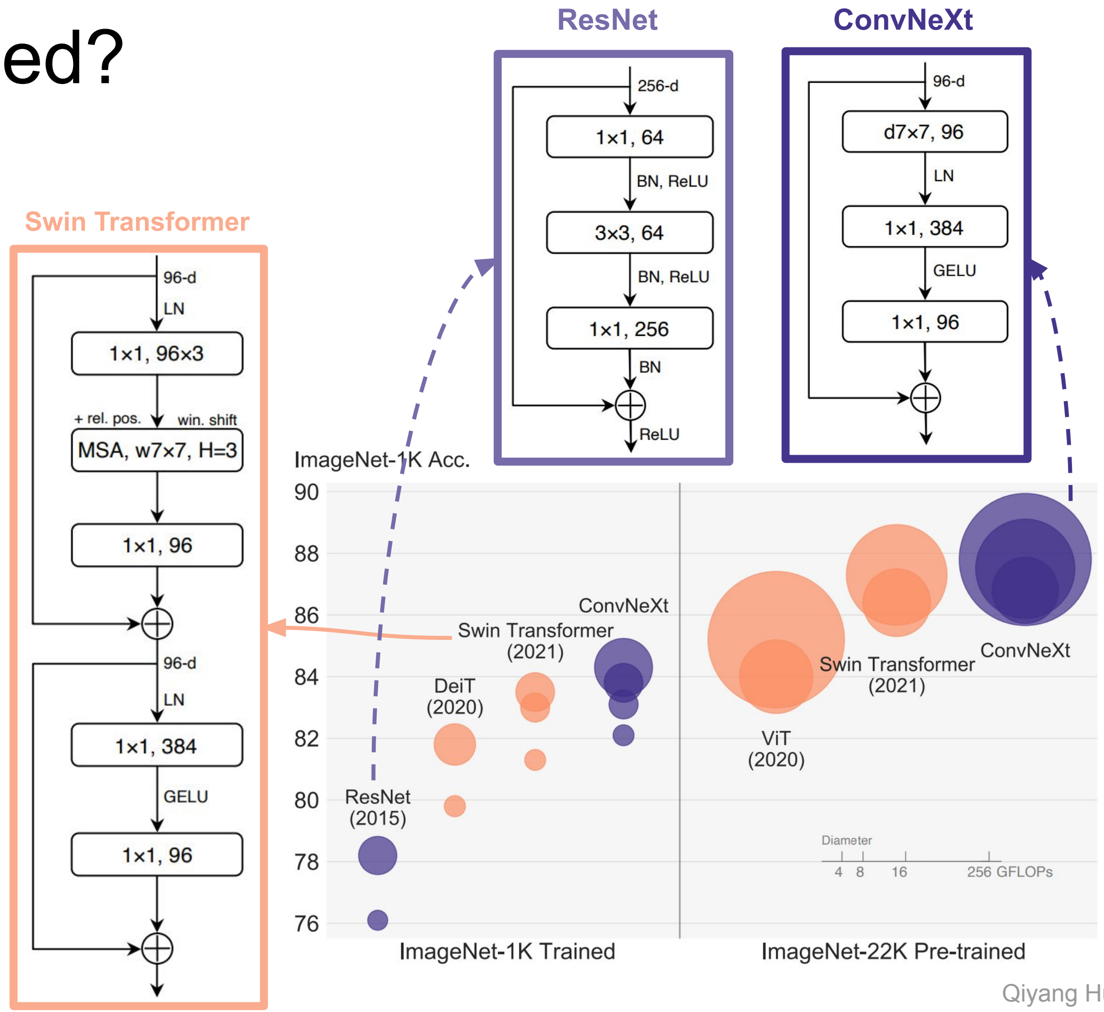
Latest Developments

- ViT & Swin-T
- ConvViT & ConvNeXt

03

ConvNet is all you need?

- Renaissance since 2012
  - AlexNet, VGGNet, ResNet, MobileNet, EfficientNet, …
  - Rapid development due to
    - “Sliding window”
    - Translation equivariance
- Challenges from NLP (2020)
  - Vision Transformers (ViT), Swin Transformers (Swin-T)
  - Replacing ConvNet by:
    - Split img to a seq of patches
    - Permutation invariant via self-att
- Revival since 2021
  - ConViT (2021), ConvNeXt (2022)
  - ConvNeXt v2 (2023): w/ self-supervised learning capabilities



Survey

[bit.ly/**3lQKSSr**](bit.ly/3lQKSSr)