Jet Propulsion Laboratory
California Institute of Technology

Machine Learning Reading Group 11/12/2020

Making Convolutional Networks Shift-Invariant Again

Richard Zhang, Adobe Research

ICML 2019

Presented by Jake Lee (398J MLIA)

With supporting materials and demos from:

What does CNN Shift Invariance Look Like? A Visualization Study

Jake Lee, Junfeng Yang, Zhangyang Wang, Columbia Univ/UT Austin

ECCV RLQ-TOD Workshop 2020

Context

- First saw paper during ICML 2019 poster session
- Briefly mentioned paper during ICML MLRG during summer 2019
- Highly relevant to CNN feature extraction phenomena I observed during internship
- Wrote workshop paper based on it during Masters, published this fall

Convolutional Neural Networks are not Shift Invariant

- Convolutional Neural Networks are popular for image classification and feature extraction
- However, transformations to the image (scaling, shifting, rotation) can significantly impact the output

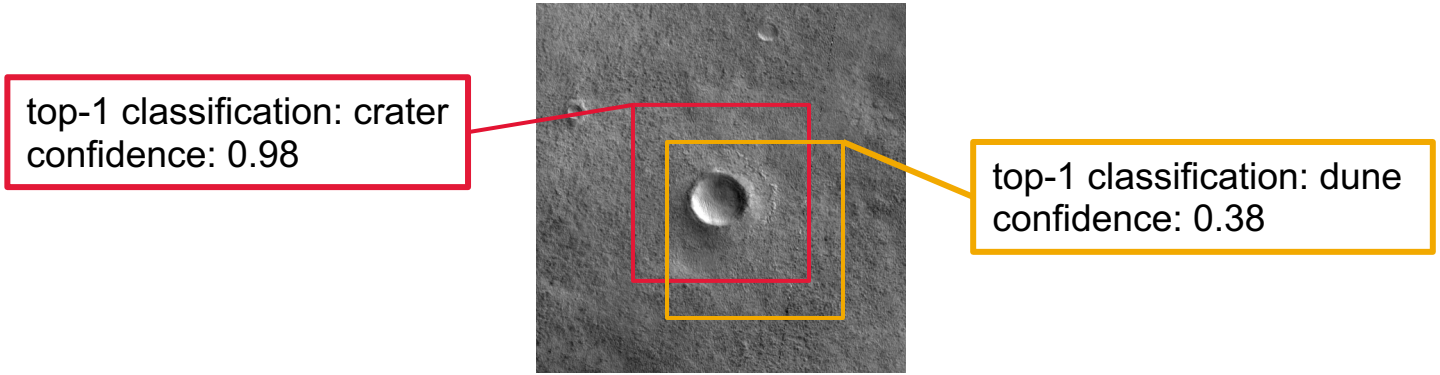
[Pei, et al. 2017]

Fig. 1. Upper row shows the original inputs. Lower row shows the violations generated by different transformations for different computer vision systems as identified by VERIVIS.

Interactive Demo: https://jakehlee.github.io/interactive-invariance/

Convolutional Neural Networks are not Shift Invariant

- Convolutional Neural Networks are popular for image classification and feature extraction
- However, transformations to the image (scaling, shifting, rotation) can hugely impact the output
- Particularly relevant for work that we often do



(hypothetical)

Naïve Solution for Shift Invariance

- Data augmentation
  - Training data could be biased, causing the model to be brittle
  - e.g. Humans are more likely to photograph objects in the center, upright
  - **Improves, but does not fix shift invariance**

from **PyTorch** off-the-shelf ImageNet training script [1]

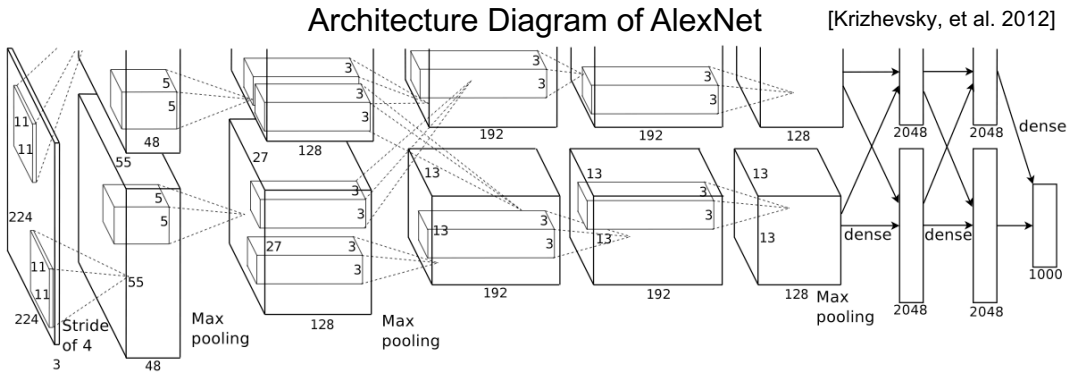
```
dataset = torchvision.datasets.ImageFolder(
    traindir,
    transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        normalize,
    ])
```

- Randomly crop image of:
  - Size 0.08~1.0x of original
  - Aspect Ratio 3:4 to 4:3
- Resize back to 224x224

- Randomly flip horizontally

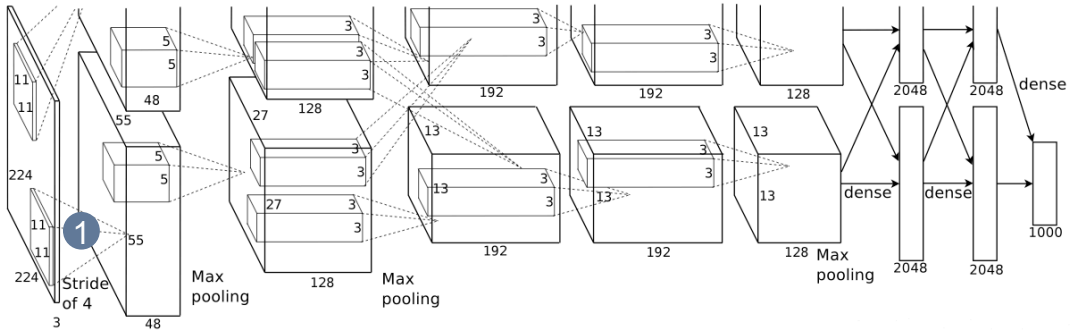
[1] https://github.com/pytorch/vision/blob/78159d61b966ec32ac330c28bea32e9e12fe6eb4/references/classification/train.py#L96-L103

A Closer Look at CNN Architecture



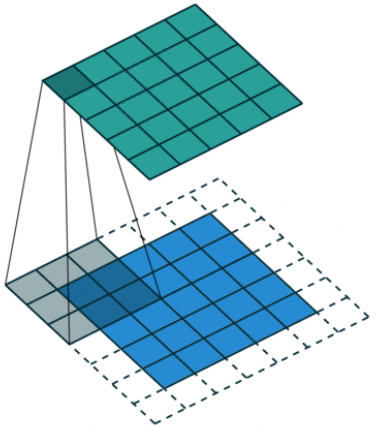
A Closer Look at CNN Architecture

Architecture Diagram of AlexNet [Krizhevsky, et al. 2012]



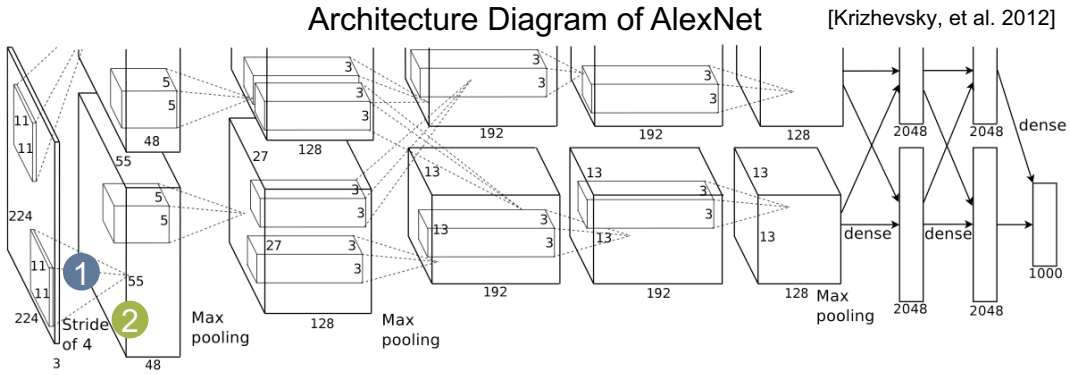
1. Filter convolution

- Weights and biases of the filters are learned

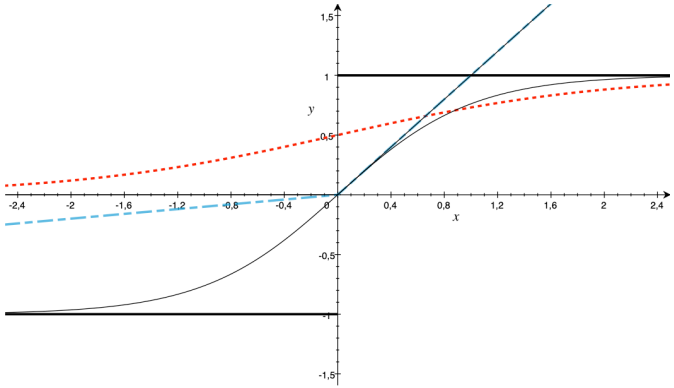


[https://github.com/vdumoulin/conv_arithmetic]

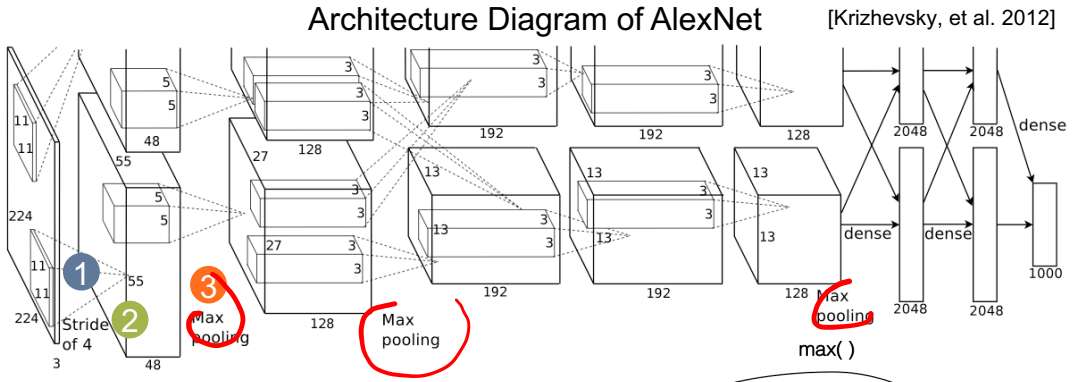
A Closer Look at CNN Architecture

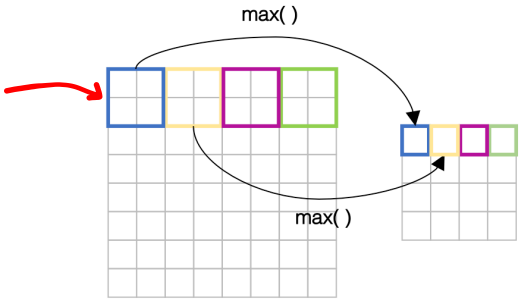


1. Filter convolution
2. **Non-linear activation function**
   - Adds nonlinearity to the model
   - otherwise just a linear perceptron



A Closer Look at CNN Architecture





1. Filter convolution
2. Non-linear activation function
3. **Max-pool downsampling**
   - Allows deeper filters to have a wider FOV w/ small filter size
   - Dimensionality reduction needed to get from 224x224x3 to 1000 classes

An Even Closer Look at Max-Pooling

- Early networks used average-pooling [LeCun et al., 1990]
  - Can be considered type of blurred downsampling
- Popular networks now use max-pooling (and variants)
  - Empirically stronger task performance [Scherer et al., 2010]

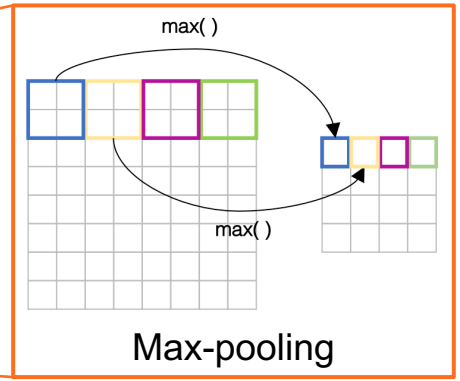
4.4 Window Functions

Small variations of the input image and shifts past the border of the pooling window can dramatically change the representation. For this reason we experimented with smoother, overlapping pooling windows. Window functions are often used to smooth an input signal in signal processing applications. We have evaluated four different window functions, as shown in Table 2.

	No overlap	Rectangular	Cone	Pyramid	Triangle	Binomial
NORB test error	5.56%	5.83%	6.29%	6.28%	10.92%	12.15%
Caltech-101 test error	52.25%	57.77%	52.36%	51.95%	69.95%	73.16%

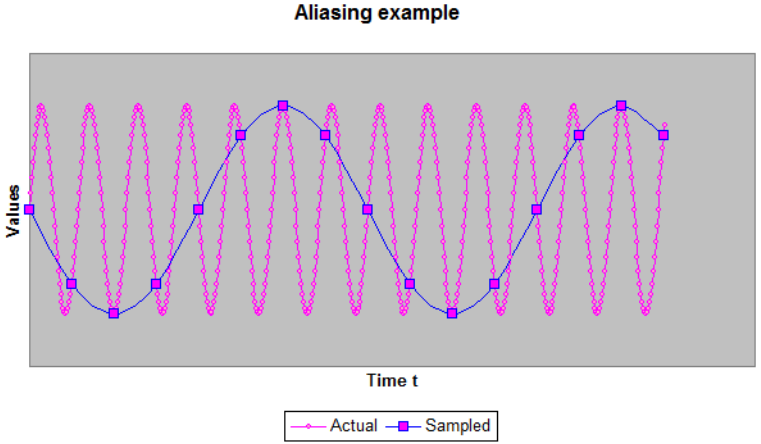
Table 2. Test error rates for NORB (after 500 epochs of training) and for Caltech-101 (after 600 epochs). Applying a window function to an overlpping neighborhood is consistently worse than using non-overlapping pooling windows.

[Scherer et al., 2010]

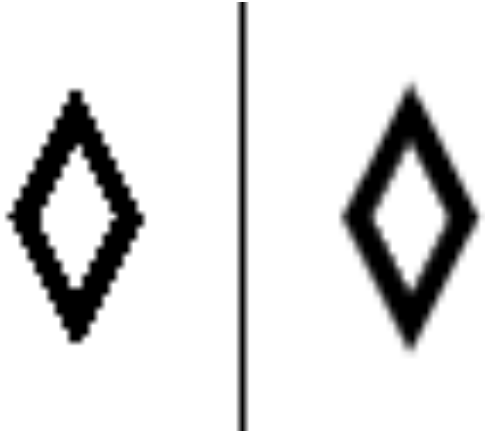


Max-Pooling Ignores the Sampling Theorem

- What is the sampling theorem (Nyquist-Shannon)?
  - When downsampling (or discretizing a continuous signal), too low of a sampling frequency causes **aliasing**
  - Fixed by sampling more or **antialiasing via blurring**



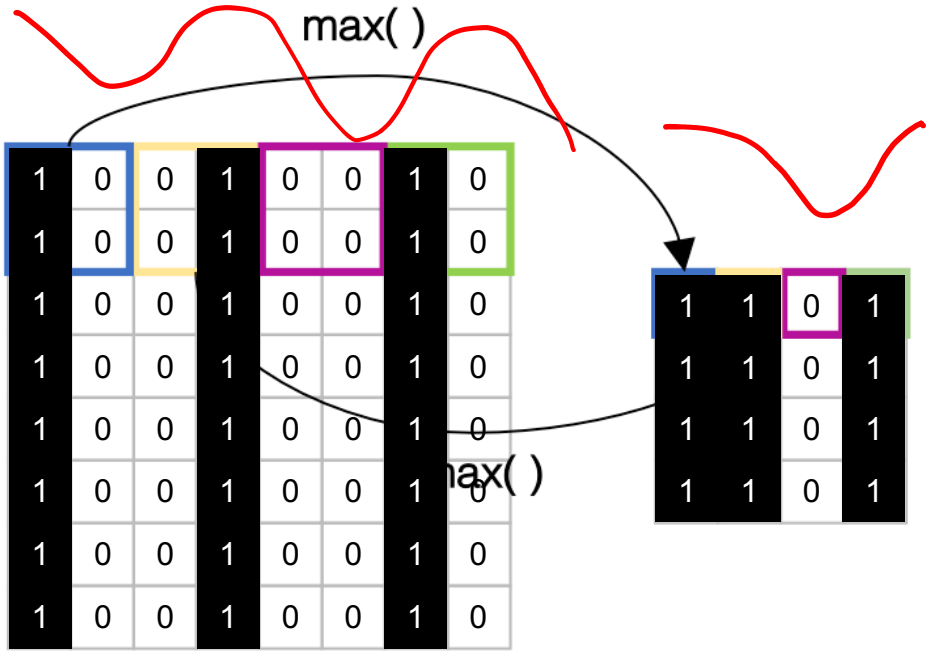
[https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/Filtering/Aliasing/aliasing.htm]



[https://en.wikipedia.org/wiki/Spatial_anti-aliasing#/media/File:Anti-aliased-diamonds.png]

Max-Pooling Ignores the Sampling Theorem

- Max-Pooling allows high-frequency noise to alias into lower frequencies



Max-Pooling Ignores the Sampling Theorem

- Known problem in 2010, but smoothed, overlapping windows impacted classification performance

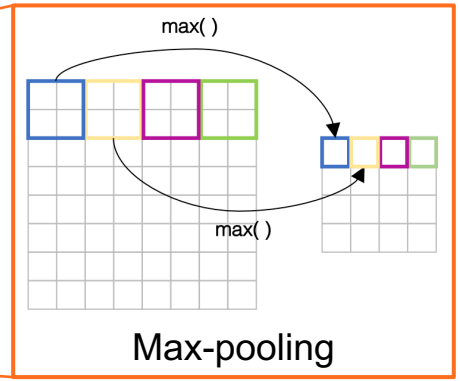
4.4 Window Functions

Small variations of the input image and shifts past the border of the pooling window can dramatically change the representation. For this reason we experimented with smoother, overlapping pooling windows. Window functions are often used to smooth an input signal in signal processing applications. We have evaluated four different window functions, as shown in Table 2.

	No overlap	Rectangular	Cone	Pyramid	Triangle	Binomial
NORB test error	5.56%	5.83%	6.29%	6.28%	10.92%	12.15%
Caltech-101 test error	52.25%	57.77%	52.36%	51.95%	69.95%	73.16%

Table 2. Test error rates for NORB (after 500 epochs of training) and for Caltech-101 (after 600 epochs). Applying a window function to an overlpping neighborhood is consistently worse than using non-overlapping pooling windows.

[Scherer et al., 2010]



So… Now What?

- Max-pooling (and variants) has been shown across several state-of-the-art method as the best downsampling method
  - … but it introduces aliasing to the signal, causing it to be brittle
- Past antialiasing implementations potentially solves this issue
  - … but it significantly reduces performance of the models

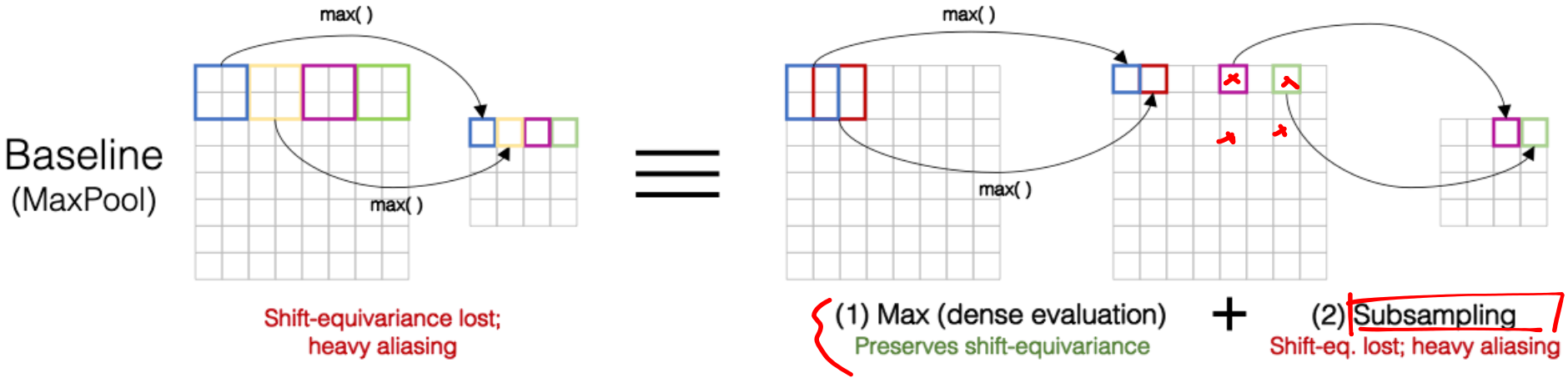
So… Now What?

- Max-pooling (and variants) has been shown across several state-of-the-art method as the best downsampling method
  - … but it introduces aliasing to the signal, causing it to be brittle
- Past antialiasing implementations potentially solves this issue
  - … but it significantly reduces performance of the models
- **Why not both?**

[Old El Paso commercial ~2009]

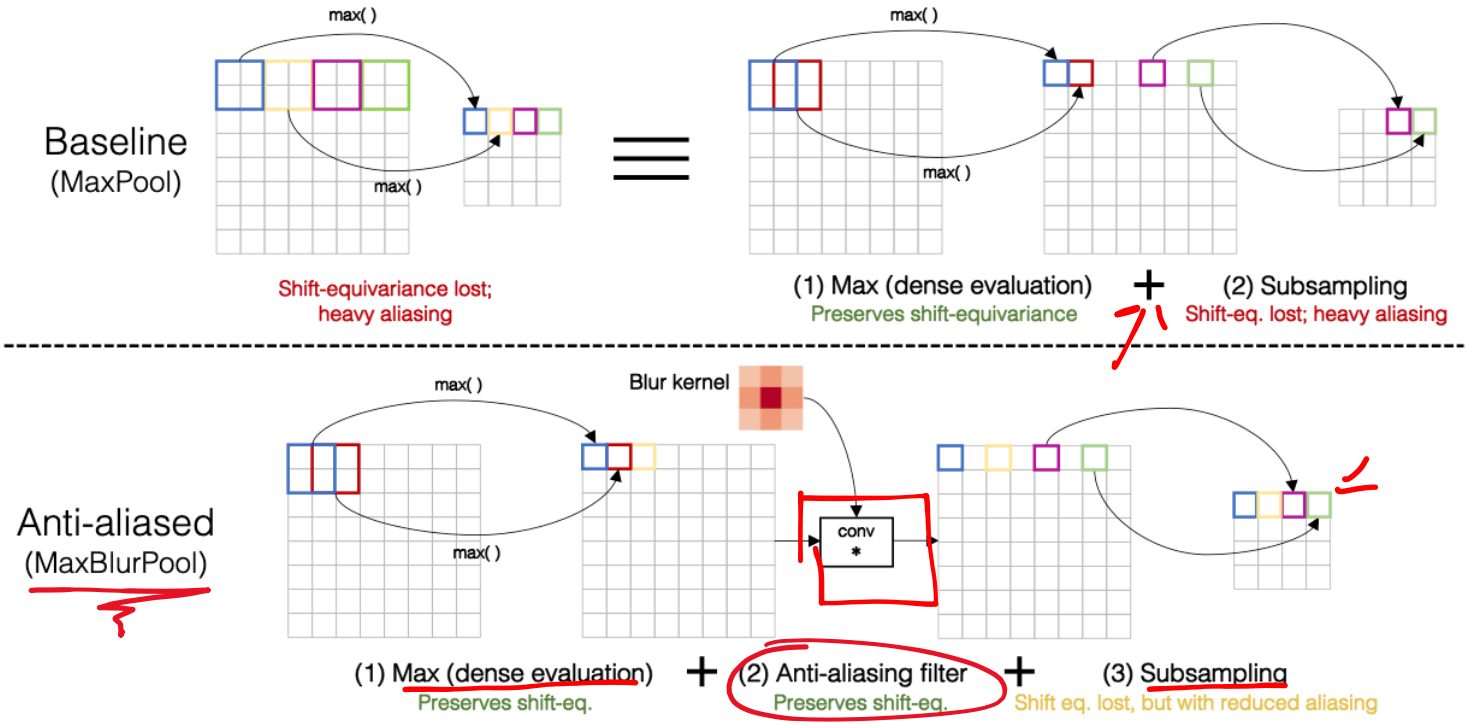
First Observation

- Max-Pooling can be deconstructed into two operations:



Second Observation

- We can squeeze in an antialiasing filter **just** before subsampling!



Does It Work?

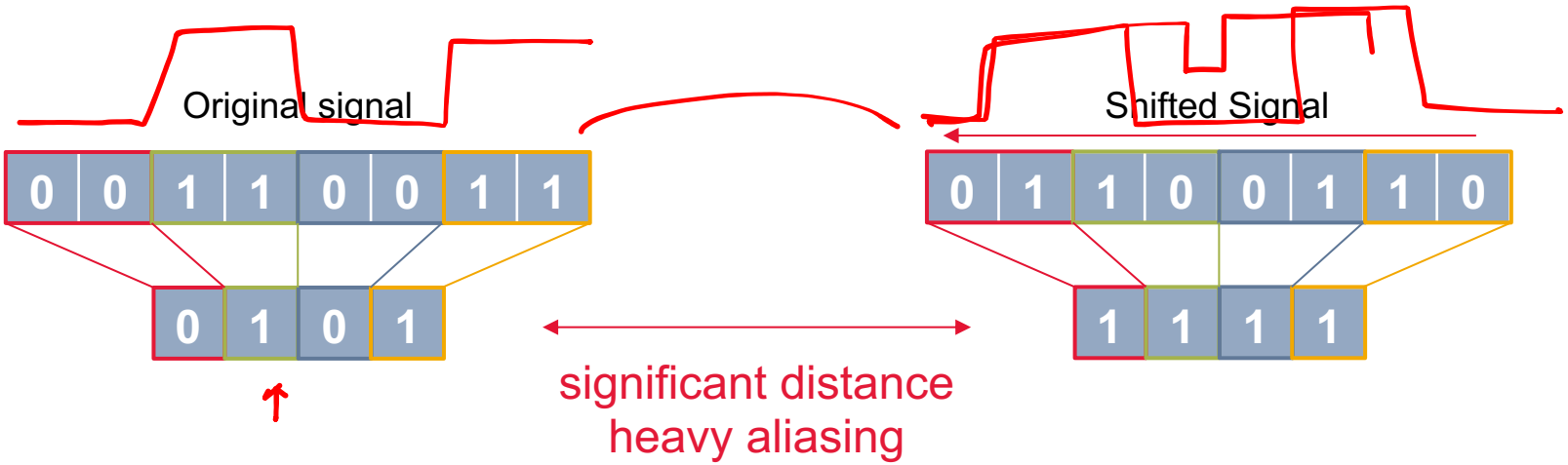
- Remember the original motivation: shift invariance

Does It Work?

- Remember the original motivation: shift invariance*
  - *The paper makes a distinction between shift **invariance** and **equivariance** but I spent too much time on the intro so I'm skipping it here, please read the paper

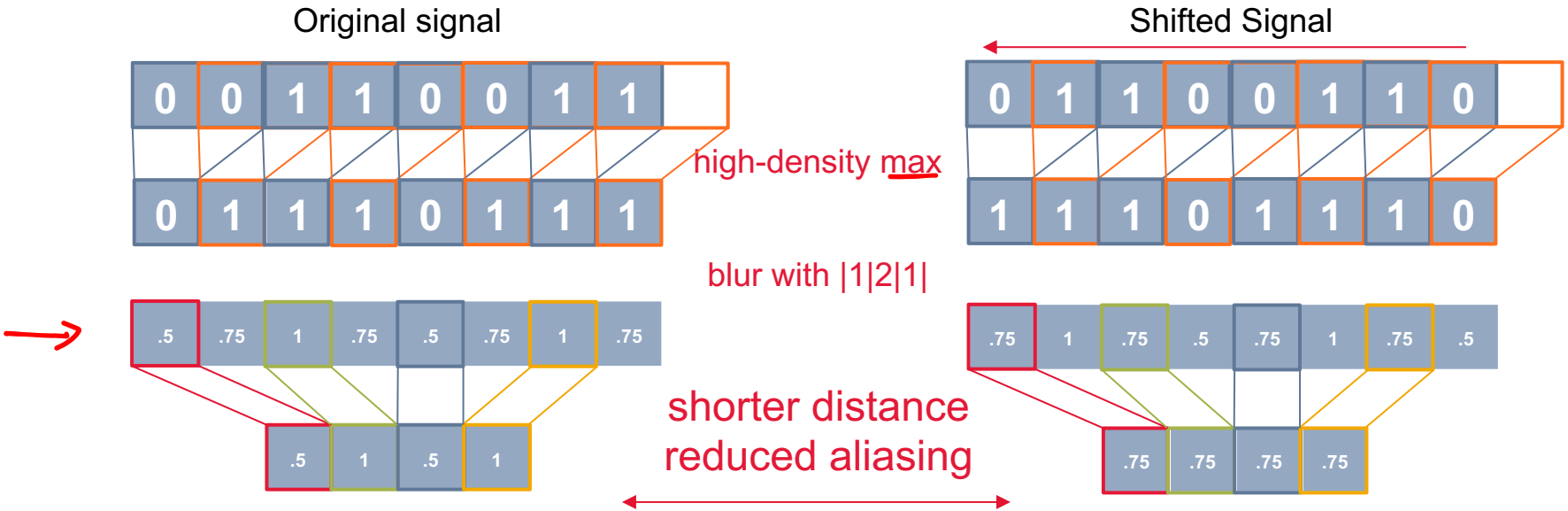
Does It Work?

- Remember the original motivation: shift invariance*
- Consider this 1D case with baseline max-pooling :



Does It Work?

- Remember the original motivation: shift invariance*
- Consider this 1D case now with **MaxBlurPool** :



Does It Work With CNN Classification?

- Before we dive into results, let's define a metric to measure invariance
- **Classification Consistency**
- $\mathbb{E}_{X,h_1,w_1,h_2,w_2}\ \mathbb{1}\{\text{argmax}\, P(\text{shift}_{h_1,w_1}(X)) = \text{argmax}\, P(\text{shift}_{h_1,w_1}(X))\}$

Does It Work With CNN Classification?

- Before we dive into results, let's define a metric to measure invariance
- **Classification Consistency**
- $\mathbb{E}_{X,h_1,w_1,h_2,w_2}\ \mathbb{1}\{\text{argmax}\,P(\text{shift}_{h_1,w_1}(X))\ = \text{argmax}\,P(\text{shift}_{h_1,w_1}(X))\}$
- "How often the network outputs the same top-1 classification, given the same image with two different shifts"
- Higher the consistency, higher the invariance

It Does Work With CNN Classification!

- Replace Max-pool (and variants) in popular off-the-shelf models and retrain on ImageNet
- **Significant** improvement in **consistency**
- **Non-trivial** improvement in **accuracy**

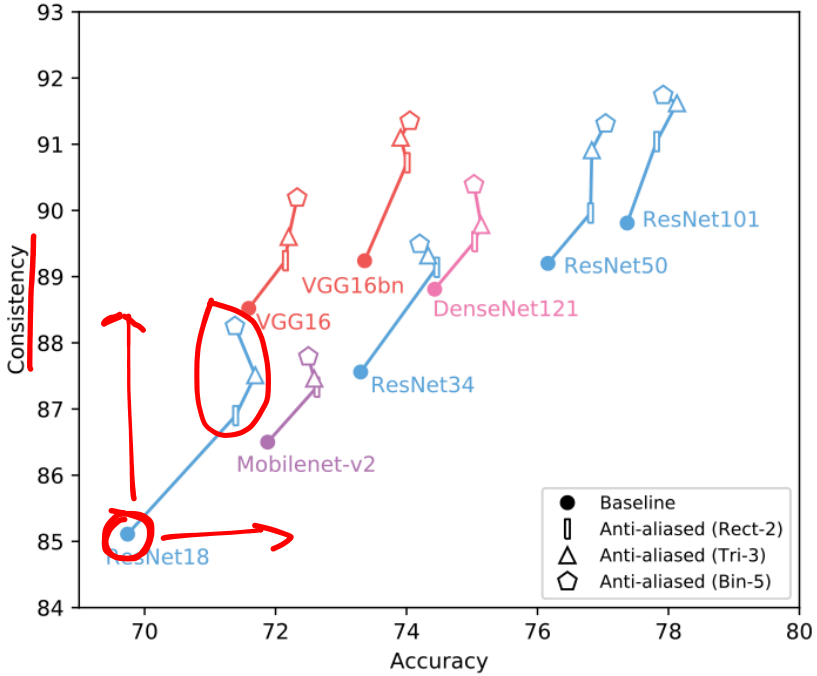


Figure 6. **ImageNet Classification consistency vs. accuracy.** Up (more consistent to shifts) and to the right (more accurate) is better. Different shapes correspond to the baseline (circle) or variants of our anti-aliased networks (bar, triangle, pentagon for length 2, 3, 5 filters, respectively). We test across network architectures. As expected, low-pass filtering helps shift-invariance. Surprisingly, classification accuracy is also improved.

It Does Work With CNN Classification!

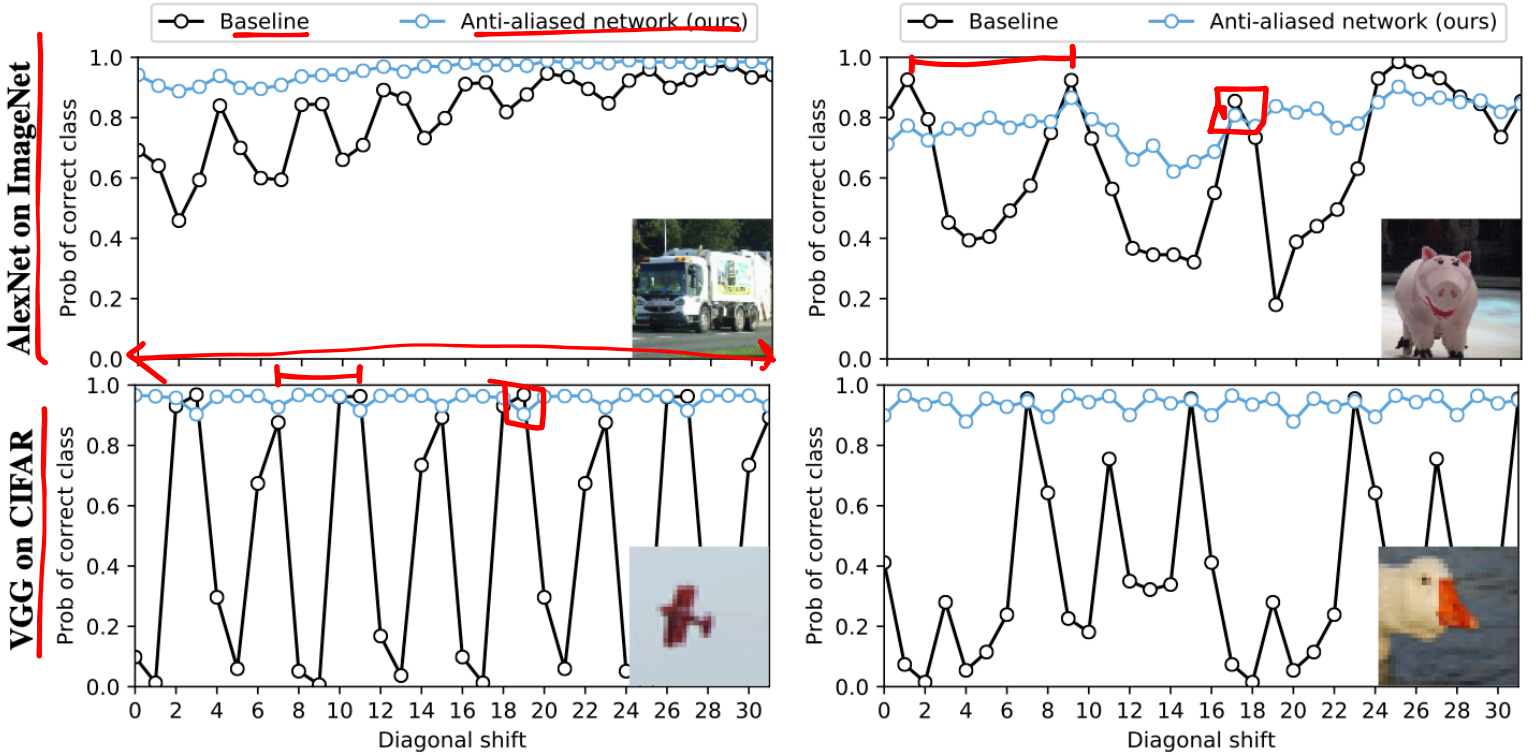


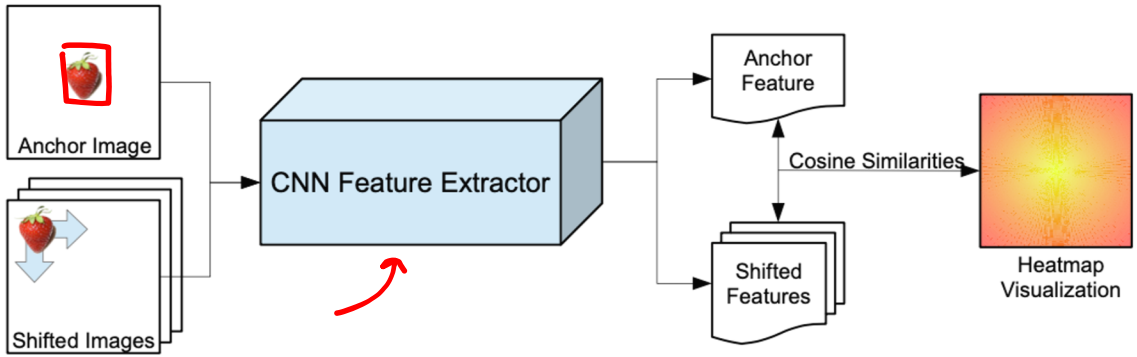
Figure 1. **Classification stability for selected images.** Predicted probability of the correct class changes when shifting the image. The baseline (black) exhibits chaotic behavior, which is stabilized by our method (blue). We find this behavior across networks and datasets. Here, we show selected examples using AlexNet on ImageNet **(top)** and VGG on CIFAR10 **(bottom)**. Code and anti-aliased versions of popular networks are available at https://richzhang.github.io/antialiased-cnns/.

Does It Work With CNN Feature Extraction?

Performed this experiment with **200** object patches shifted to generate **4.5 million images**

Feature Similarity Experiment

1. Exhaustively shift an object image patch across a white background
2. Extract features from all shifted images
3. Calculate cosine similarities between all shifted features and an “anchor”



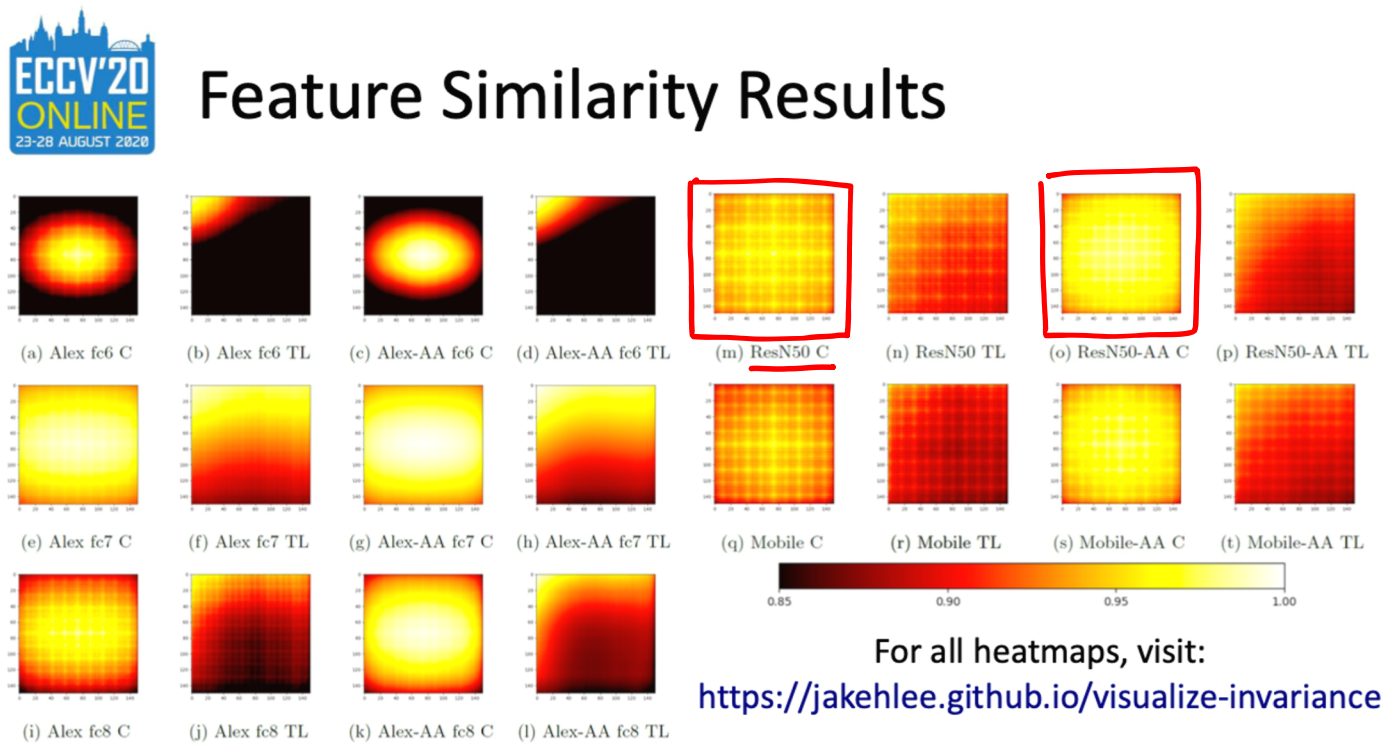
Lee, et al. - ECCV RLQ 2020

7

It Stabilizes CNN Feature Extraction!

Improved **local invariance,** but not **global invariance**

*Related to invariance vs equivariance



Interactive Demo: https://jakehlee.github.io/interactive-invariance/

Can I Use This?

- Yes! … if you're using PyTorch ☺

Can I Use This?

- Yes! … if you're using PyTorch ☺
  - Please stop using Caffe ☹

Can I Use This?

- Yes! … if you're using PyTorch ☺
  - Please stop using caffe ☹
- ```
  $ pip install antialiased-cnn
  ```
  - Package source https://github.com/adobe/antialiased-cnns
  - CC BY-NC-SA 4.0 License
  ```

Can I Use This?

- Yes! … if you're using PyTorch ☺
  - Please stop using caffe ☹
- `$ pip install antialiased-cnn`
  - Package source https://github.com/adobe/antialiased-cnns
  - CC BY-NC-SA 4.0 License
- Models pre-trained on ImageNet
  - ```
    import antialiased_cnns
    model = antialiased_cnns.resnet50(pretrained=True)
    ```
- Layers for model building
  - ```
    C = 10 # example feature channel size
    blurpool = antialiased_cnns.BlurPool(C, stride=2) # BlurPool layer; use to downsample a feature ma
    ex_tens = torch.Tensor(1,C,128,128)
    print(blurpool(ex_tens).shape) # 1xCx64x64 tensor
    ```

Conclusion

- Correctly adding antialiasing to CNNs significantly improved consistency **and** accuracy
- Don't ignore fundamentals in search for performance
  - Lemma: don't ignore a foundational theorem of signal processing when processing signals
- Use this work as an excuse to **modernize** your Deep Learning pipelines
  - Probably should stop using AlexNet/CaffeNet
- Future Work
  - Further downstream improvements: GANs, NN Retrieval, video, anti-adversarial robustness, detection/segmentation
  - Further investigate generalization