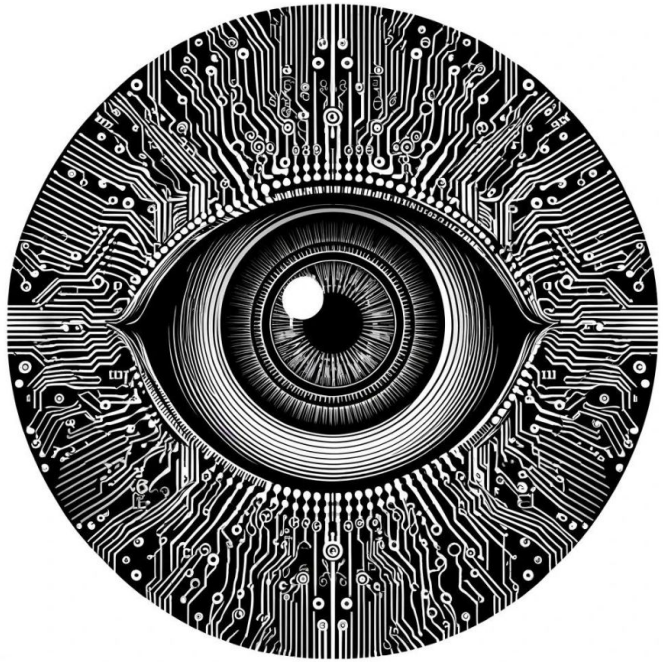
KI Service Zentrum by Hasso-Plattner-Institut HPI

Normalizing Input Values and Inference with Pretrained Models



Antonio Rueda-Toicen

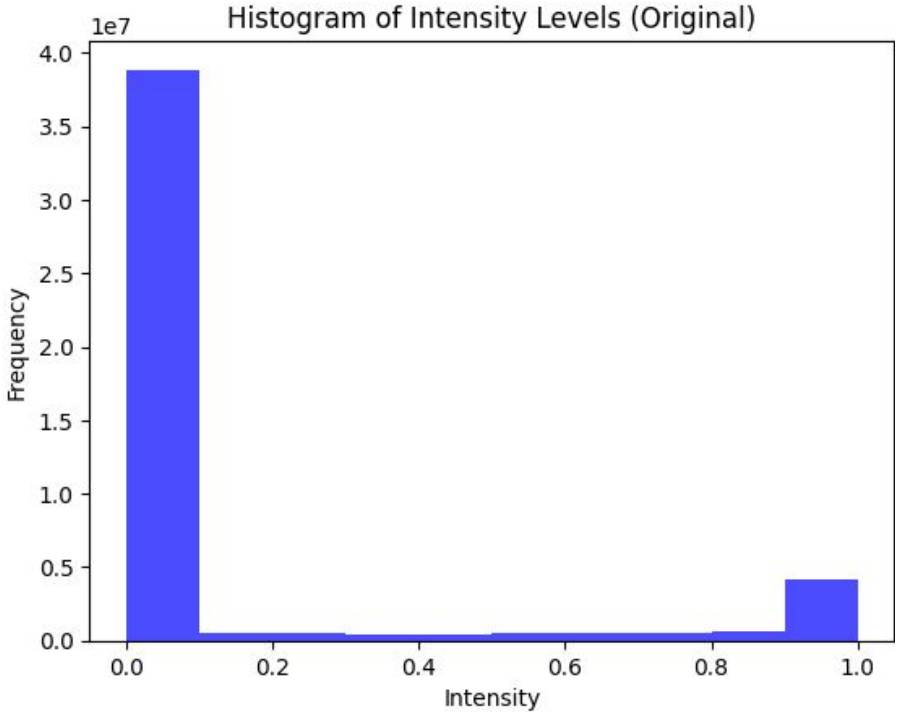
SPONSORED BY THE
Federal Ministry of Education and Research

Learning goals

- Evaluate the effect of normalizing input values using statistics from the training set
- Describe effect of input normalization on the output of models pretrained on the Imagenet dataset

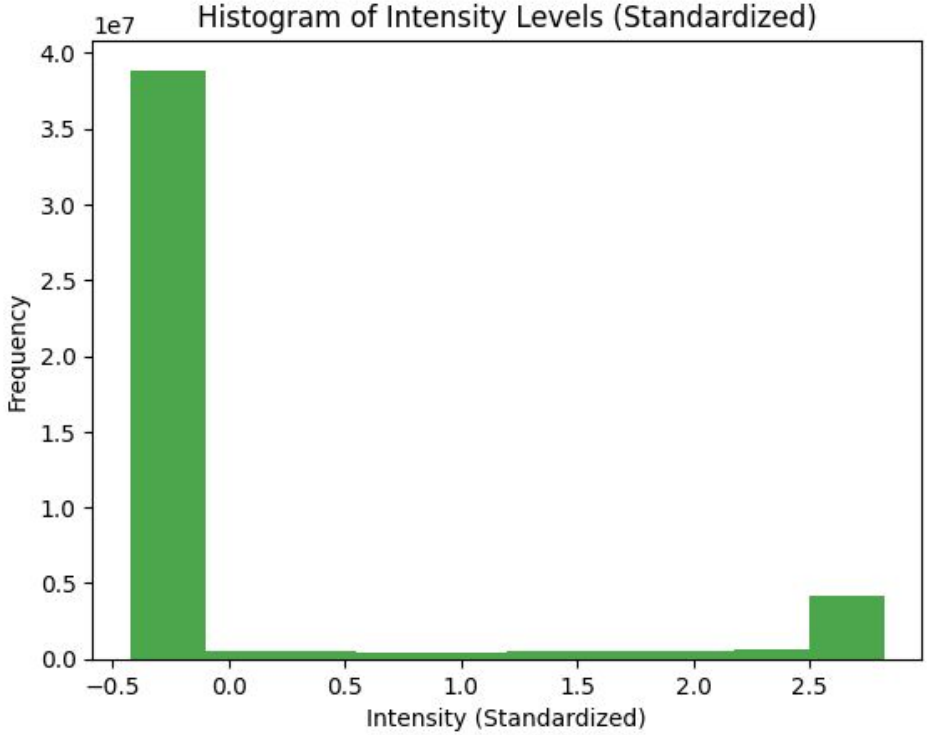
What is input scaling?

- Transforming input values to a range different than the original while preserving the shape of the original distribution
- The standard scaler centers the distribution on zero while making the variance equal to one



$$\mu = 0.1307, \quad \sigma = 0.3081$$

$$z_i = \frac{x_i - \mu}{\sigma}$$



$$\mu = 0.0, \quad \sigma = 1.0$$

Why do we do scaling?

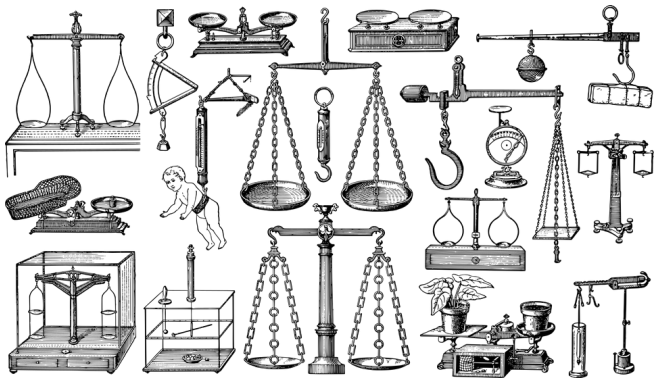
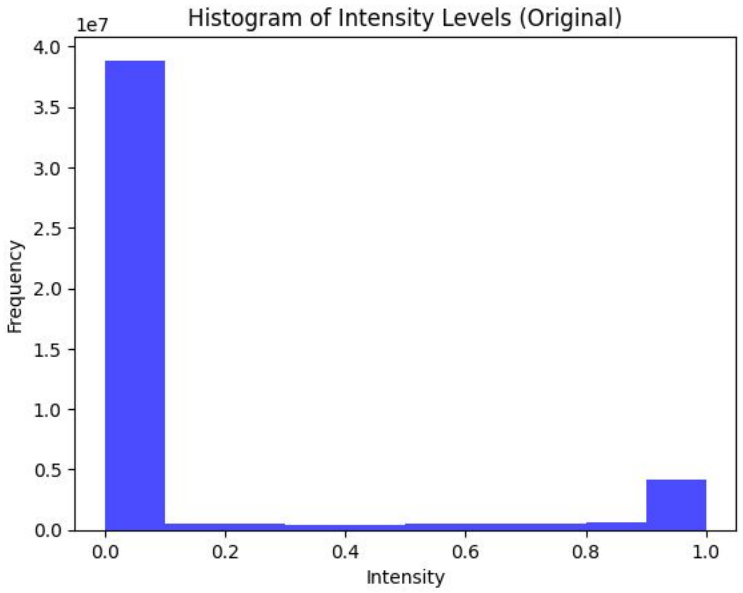
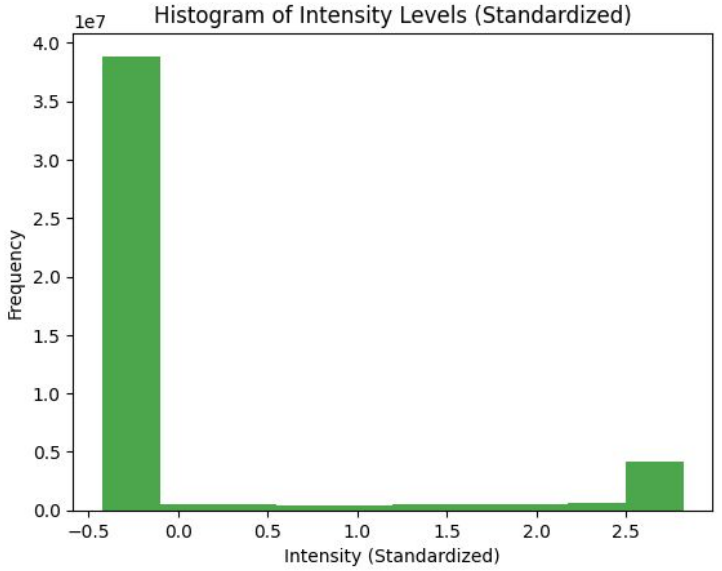


Image from Pexels

1. Has been found in experiments to speed up training.
2. We benefit from using it whenever we use a pretrained model that used it during training.



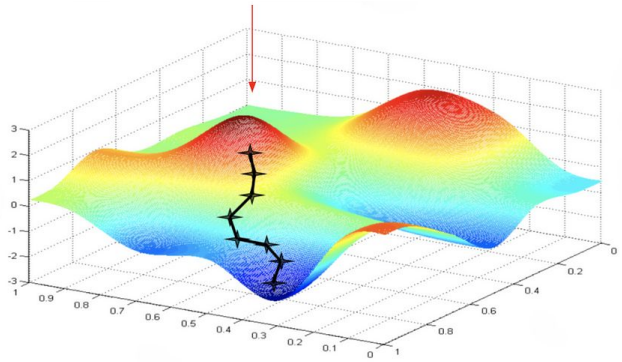
$$z_i = \frac{x_i - \mu}{\sigma}$$



4.3 Normalizing the Inputs

Convergence is usually faster if the average of each input variable over the training set is close to zero. To see this, consider the extreme case where all the inputs are positive. Weights to a particular node in the first weight layer are updated by an amount proportional to δx where δ is the (scalar) error at that node and x is the input vector (see equations (5) and (10)). When all of the components of an input vector are positive, all of the updates of weights that feed into a node will be the same sign (i.e. sign(δ)). As a result, these weights can only all decrease or all increase *together* for a given input pattern. Thus, if a weight vector must change direction it can only do so by zigzagging which is inefficient and thus very slow.

In the above example, the inputs were all positive. However, in general, any shift of the average input away from zero will bias the updates in a particular direction and thus slow down learning. Therefore, it is good to shift the inputs so that the average over the training set is close to zero. This heuristic should be applied at all layers which means that we want the average of the *outputs* of a node to be close to zero because these outputs are the inputs to the next layer [19]. This problem can be addressed by coordinating how the inputs are



Excerpt from '[Efficient Backprop: Tricks of the Trade](#)' by LeCun, Bottou, Orr, and Müller (1998)

The standard scaler for normalization

Mean (μ):

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Standardized Value (z_i):

$$z_i = \frac{x_i - \mu}{\sigma}$$

Standard Deviation (σ):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

```
computed_mean = torch.mean(train_dataset_elements)
computed_std  = torch.std(train_dataset_elements)

normalization_transform = transforms.Normalize(
                            (computed_mean),
                             (computed_std)
                          )
```

The Imagenet dataset

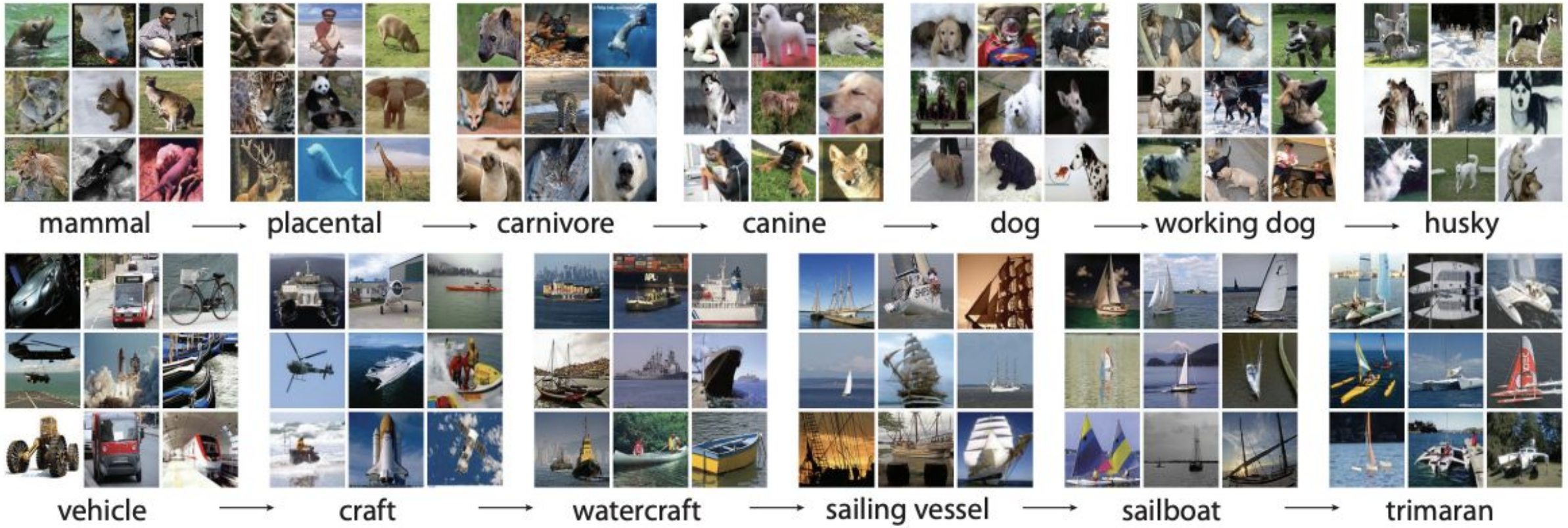
14,197,122 images, 21841 synsets indexed

Home Download Challenges About

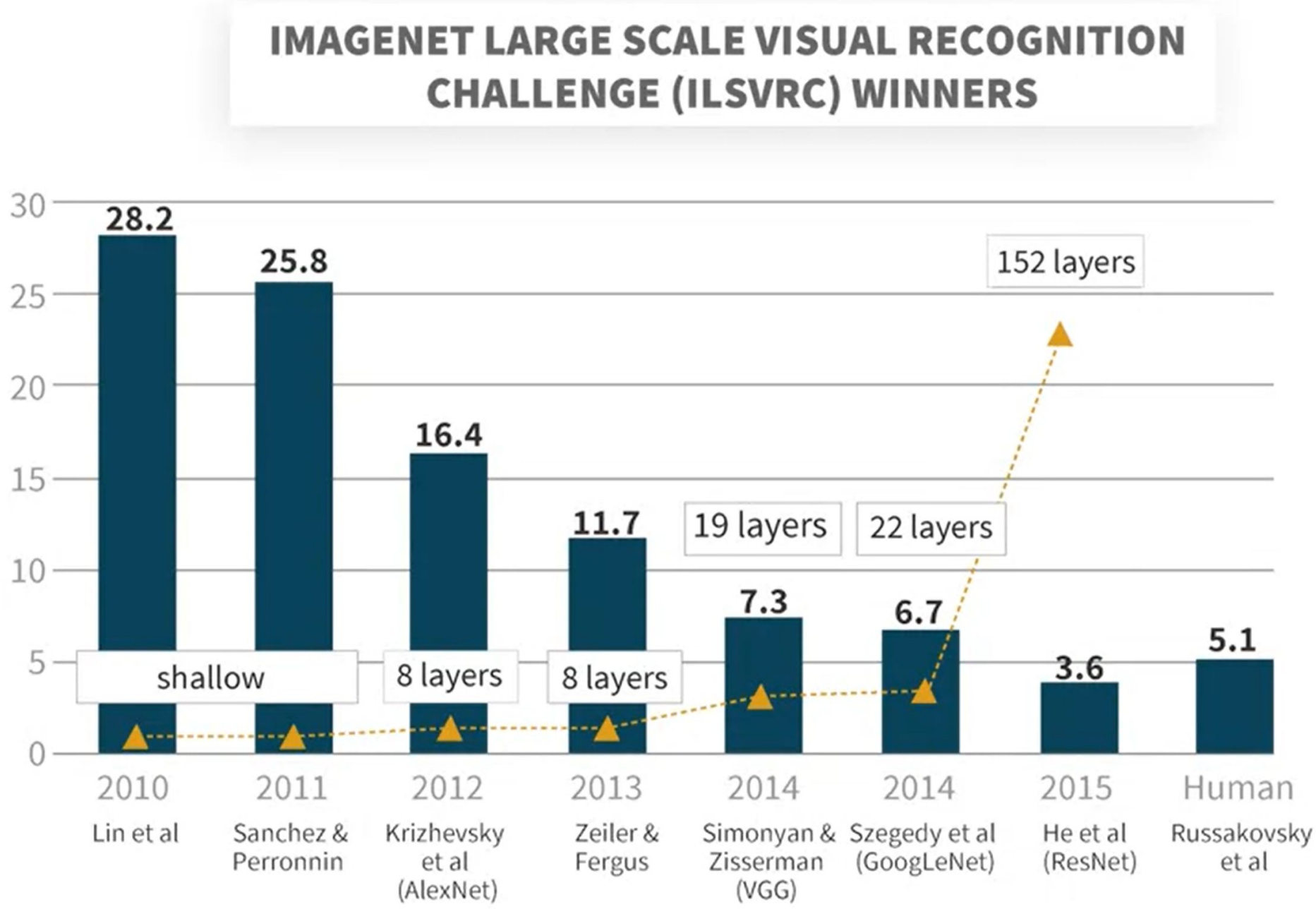
Not logged in. Login I Signup

ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been instrumental in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

https://www.image-net.org/

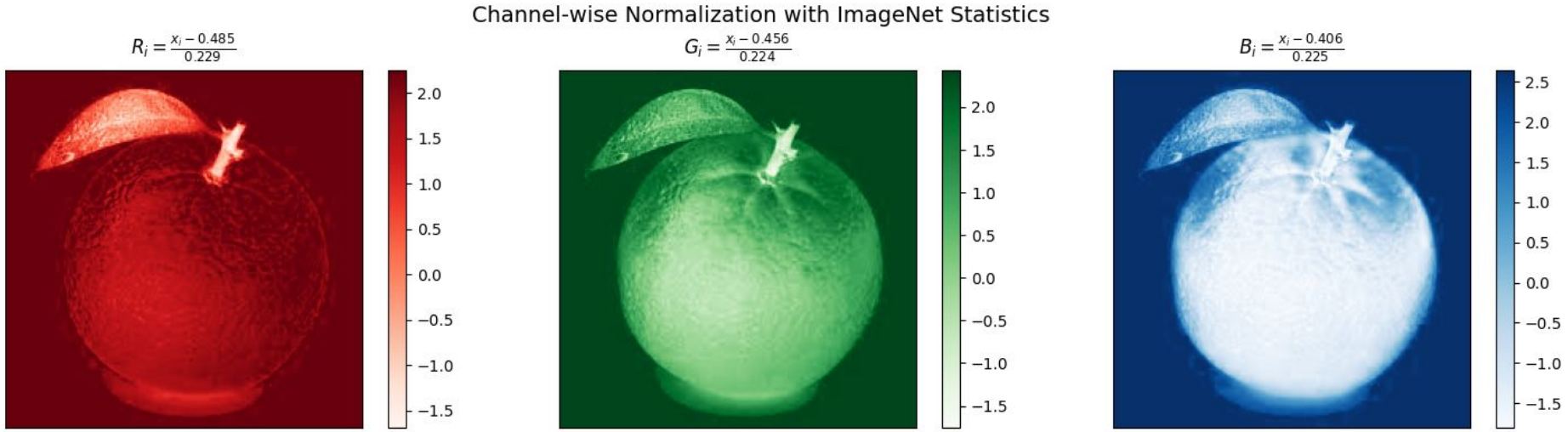


Source: Introduction to Imagenet

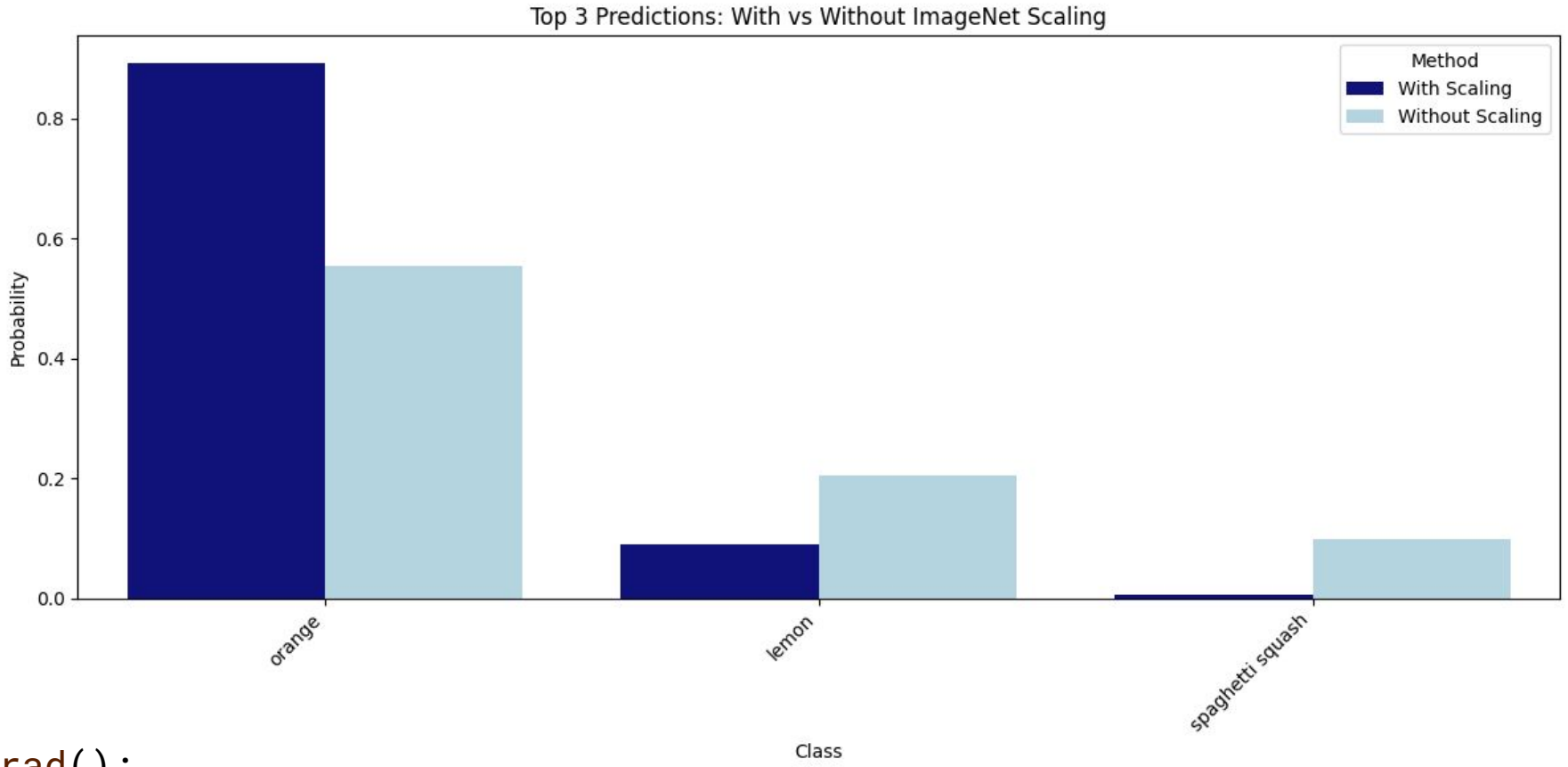

IMAGENET LARGE SCALE VISUAL RECOGNITION CHALLENGE (ILSVRC) WINNERS
30
25
20
15
10
5
0
28.2
25.8
16.4
11.7
7.3
6.7
3.6
5.1
shallow
8 layers
8 layers
19 layers
22 layers
152 layers
2010
Lin et al
2011
Sanchez & Perronnin
2012
Krizhevsky et al (AlexNet)
2013
Zeiler & Fergus
2014
Simonyan & Zisserman (VGG)
2014
Szegedy et al (GoogLeNet)
2015
He et al (ResNet)
Human
Russakovsky et al

The Imagenet transformations

```
# ImageNet scaling transformation
        transform = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
              # This will convert the PIL image to a torch tensor with the same uint8 type
            transforms.ToImage(),
            # We convert the uint8 tensor to a float32 tensor and divide by 255
            transforms.ToDtype(torch.float32, scale=True),
            # We apply the standard scaler
            transforms.Normalize(
                mean=[0.485, 0.456, 0.406], #Red, Green, Blue
                std =[0.229, 0.224, 0.225]  #These stats come from the Imagenet's training set
            )
        ])
```



Predicting with and without scaling




```
with torch.no_grad():
      output = model(img_tensor)

 probabilities = torch.nn.functional.softmax(output[0], dim=0)
 top3_prob, top3_idx = torch.topk(probabilities, 3)
```

Accessing the transformations from torchvision's pretrained models

```python
import torchvision.transforms as transforms
from torchvision.models import ConvNeXt_Tiny_Weights

model =
torchvision.models.convnext_tiny(weights="IMAGENET1K_V1")

# Access the weights object
weights = ConvNeXt_Tiny_Weights.IMAGENET1K_V1

# Get the transform functions from the weights object
transforms_train = weights.transforms()
transforms_train
```


```
ImageClassification(
    crop_size=[224]
    resize_size=[236]
    mean=[0.485, 0.456, 0.406]
    std=[0.229, 0.224, 0.225]
    interpolation=InterpolationMode.BILINEAR
)
```

Summary

Input normalization speeds up training

- Centers data around zero (typically mean = 0)
- Scales features to have similar variances (often standard deviation = 1)

ImageNet preprocessing pipeline

- Includes resizing, center cropping, and using Imagenet's training set statistics for mean and standard deviation of its training set of about 1 million images.

Impact on model performance and pretraining

- Proper scaling improves model predictions, particularly when using pre-trained models
- PyTorch provides easy access to the transformation that was used to train a model

SPONSORED BY THE
Federal Ministry of Education and Research

Further reading and references

Efficient Backprop

- https://cseweb.ucsd.edu/classes/wi08/cse253/Handouts/lecun-98b.pdf

An Introduction to Imagenet

- https://blog.roboflow.com/introduction-to-imagenet/

Documentation of Convnext on PyTorch

- https://pytorch.org/vision/main/models/generated/torchvision.models.convnext_base.html#torchvision.models.convnext_base

SPONSORED BY THE
Federal Ministry of Education and Research