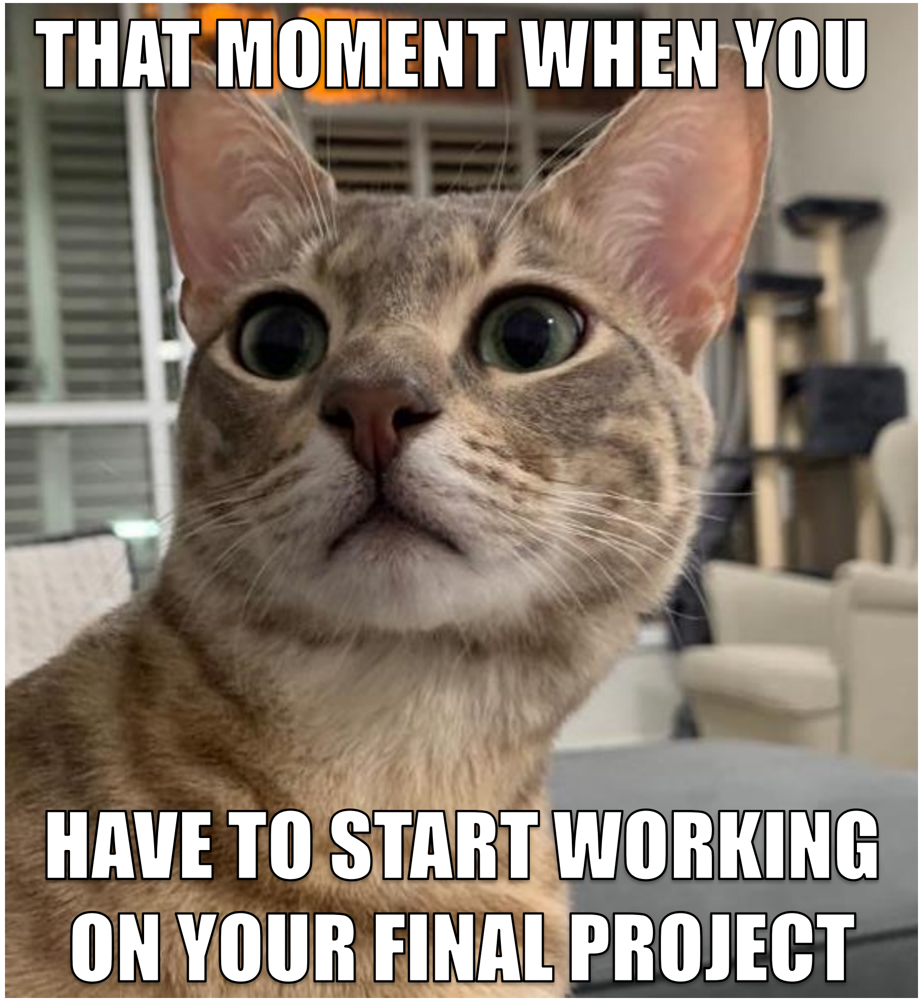
[redacted]

Deep Learning Practitioner's Toolbox

[redacted]


THAT MOMENT WHEN YOU
HAVE TO START WORKING
ON YOUR FINAL PROJECT

[redacted]

The Toolbox

- Using Pretrained Models
- Hooks
- Monitoring Experiments
- Transforms
- Reproducibility
- Saving & Loading Models

[redacted]

Don’t Reinvent the Wheel!

Use Existing Tools!

Pretrained Models

[redacted]

PyTorch
HUB

timm

Hugging Face

[redacted]

Timm

```
!pip install timm
import timm
import torch

# list of available models in pyTorch's repo.
timm.list_models(pretrained=True)

# printed: ['adv_inception_v3', 'cait_m36_384', 'cait_m48_448', ...]
# ~450 models

# load model
model = timm.create_model('resnet18', pretrained=True)
model.eval()

input = torch.randn(1, 3, 224, 224)
output = model(input)

# output.shape is 1x1000
```

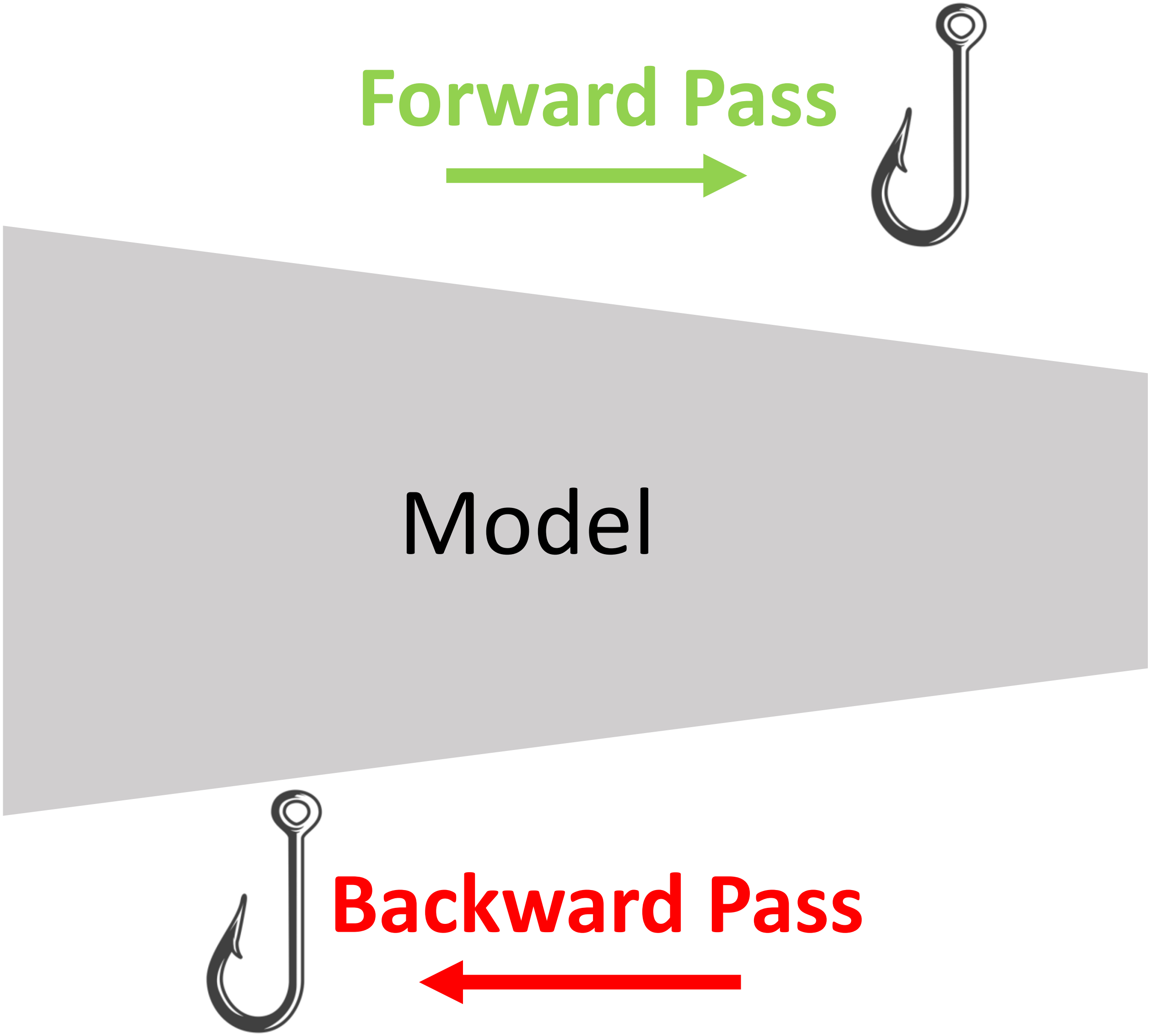
[redacted]

Torch Hub

```
import torch

# list of available models in pyTorch's repo.
torch.hub.list('pytorch/vision')

# printed: ['alexnet', 'deeplabv3_mobilenet_v3_large’,
'deeplabv3_resnet101’, ...]

# load model
model = torch.hub.load('pytorch/vision', 'resnet18', pretrained=True)
model.eval()

input = torch.randn(1, 3, 224, 224)
output = model(input)

# output.shape is 1x1000
```

[redacted]

Hooks



Forward Pass

Model

Backward Pass

Register a Hook

```
def my_hook(module, input, output):
  # module: the module being hooked
  # input: a tuple of inputs
  # output: a tuple of outputs
  print("hook!", input[0].shape, output[0].shape)

# register the hook
net.conv1.register_forward_hook(my_hook)

# use the hook
y = net(x)
# printed: "hook! torch.Size([...]) torch.Size([...])"
```

[redacted]

Remove a Hook

- A remove handle is returned during the registration.

```
# register the hook
handle = net.conv1.register_forward_hook(my_hook)

# remove the hook
handle.remove()

# use the hook
y = net(x)
# nothing is printed
```

[redacted]

Feature Extraction

```
class FeatureExtractor:
  def __init__(self, model):
    self._feature = None
    self.model = model
    self._handle = self.model.layer1.register_forward_hook(self._get_feature_hook())

  def __del__(self):
    self._handle.remove()

  def _get_feature_hook(self):
    def _get_feature(module, input, output):
      self._feature = output
    return _get_feature

  def __call__(self, input):
    output = self.model(input)
    return self._feature
```

[redacted]

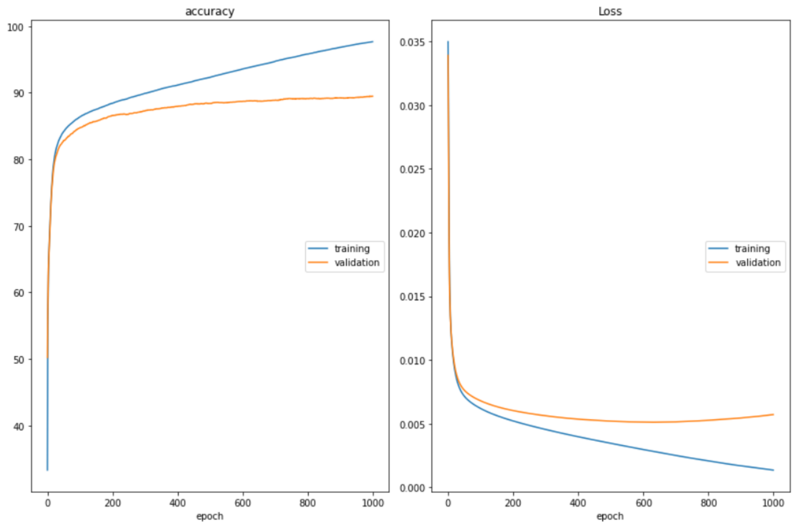
Feature Extraction

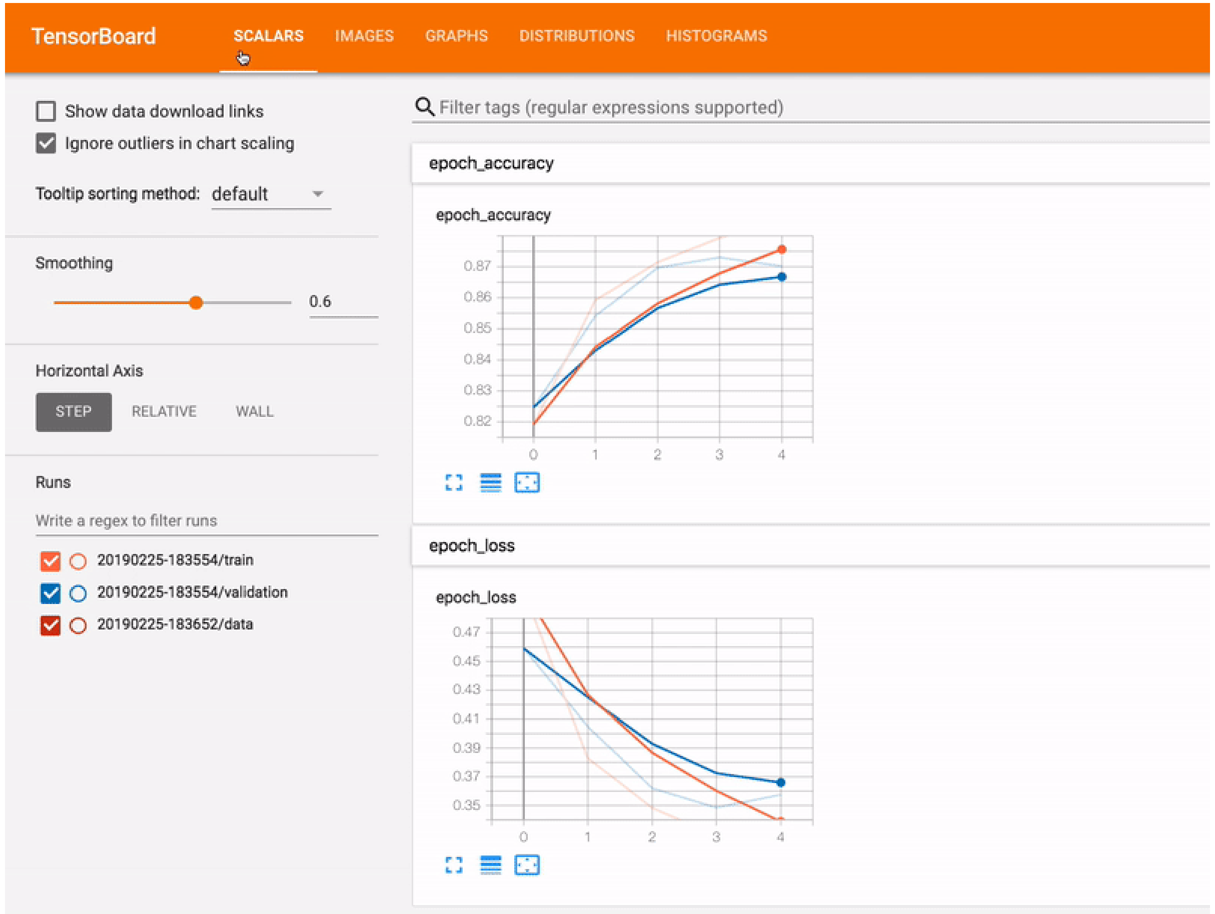
```
# create a model
model = torch.hub.load('pytorch/vision', 'resnet18', pretrained=True)

# create a feature extractor
feature_extractor = FeatureExtractor(model)

# create input
input = torch.randn(1, 3, 224, 224)

# get features
feat = feature_extractor(input)

# feat.shape is 1x64x56x56
```

[redacted]

TensorBoard

live loss plot





Filter tags (regular expressions supported)

☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ 20190225-183554/train

☑ ○ 20190225-183554/validation

☑ ○ 20190225-183652/data

Image credit to Tensorflow

[redacted]

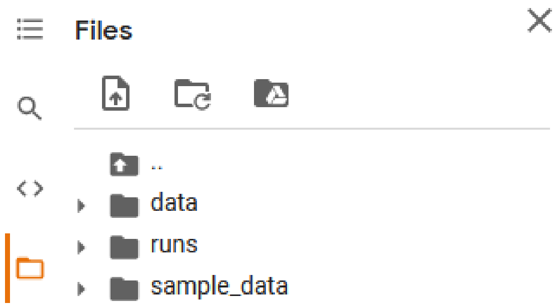
TensorBoard Preparations

```
# install tensorboard
!pip install tensorboard

%load_ext tensorboard

...

# run the user interface
%tensorboard --logdir .
```

TensorBoard Summary Writer

[redacted]

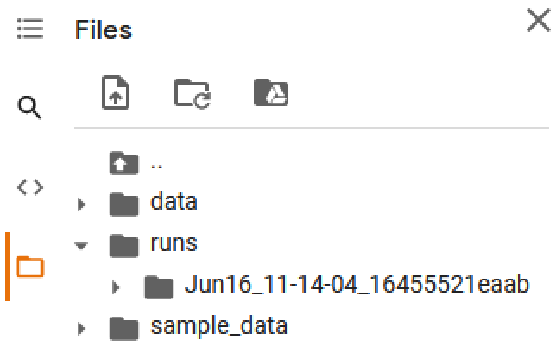
```
from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter()
```



[redacted]

TensorBoard Summary Writer

```
from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter()
```



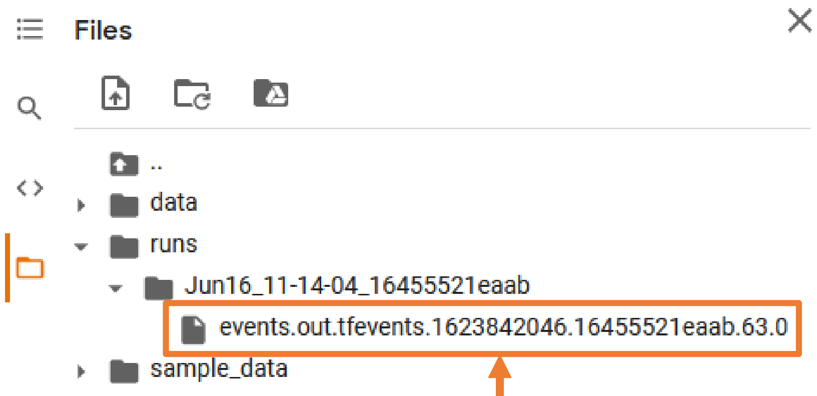
Files

..
data
runs
Jun16_11-14-04_16455521eaab
sample_data

[redacted]

TensorBoard Summary Writer

```
from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter()
```



Files

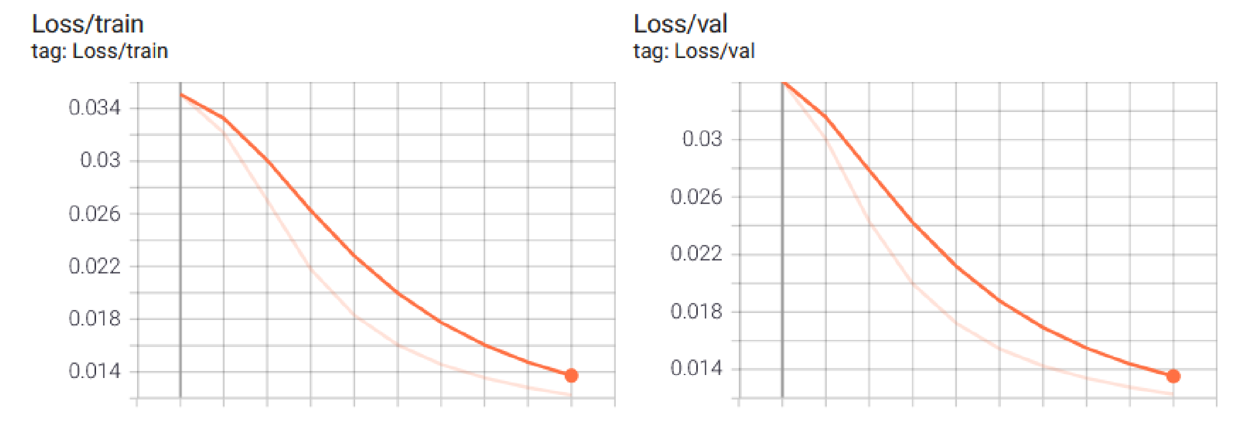
```
..
data
runs
  Jun16_11-14-04_16455521eaab
    events.out.tfevents.1623842046.16455521eaab.63.0
sample_data
```

Everything is here!

[redacted]

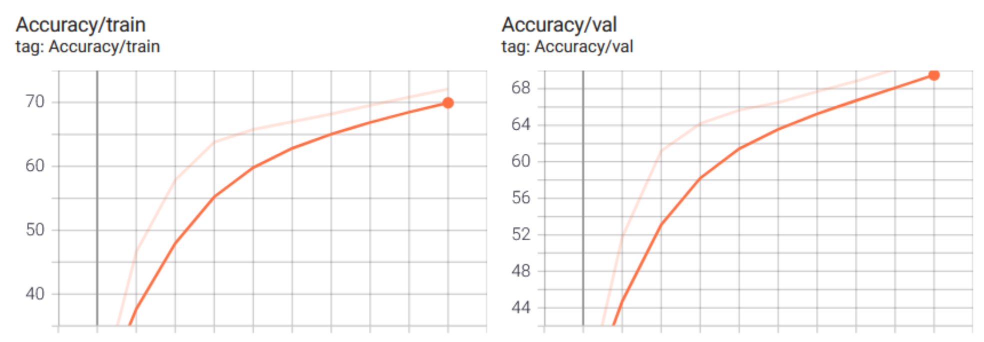
TensorBoard Summary Writer

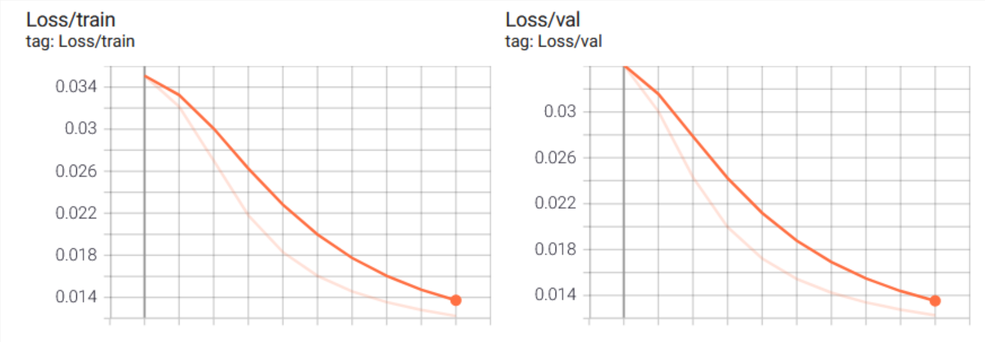
```
# train
for epoch in range(EPOCHS):
    ...
    writer.add_scalar("Loss/train", loss, epoch)
    writer.add_scalar("Loss/val", val_loss, epoch)
    ...
```



TensorBoard Summary Writer

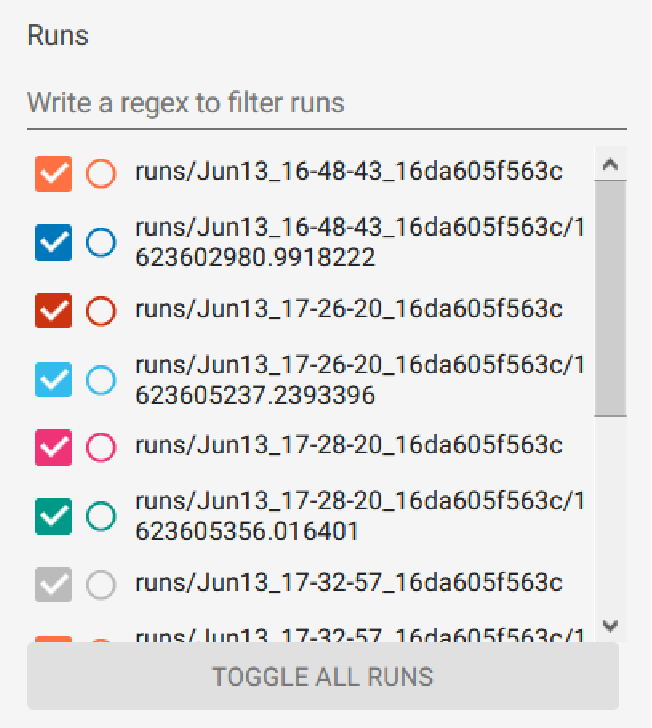
```
from torch.utils.tensorboard import SummaryWriter

writer = SummaryWriter()

# train
for epoch in range(EPOCHS):
  ...
  writer.add_scalar("Loss/train", loss, epoch)
  writer.add_scalar("Loss/val", val_loss, epoch)
  writer.add_scalar(“Accuracy/train", acc, epoch)
  writer.add_scalar(“Accuracy/val", val_acc, epoch)
  ...
```





TensorBoard Summary Writer

[redacted]

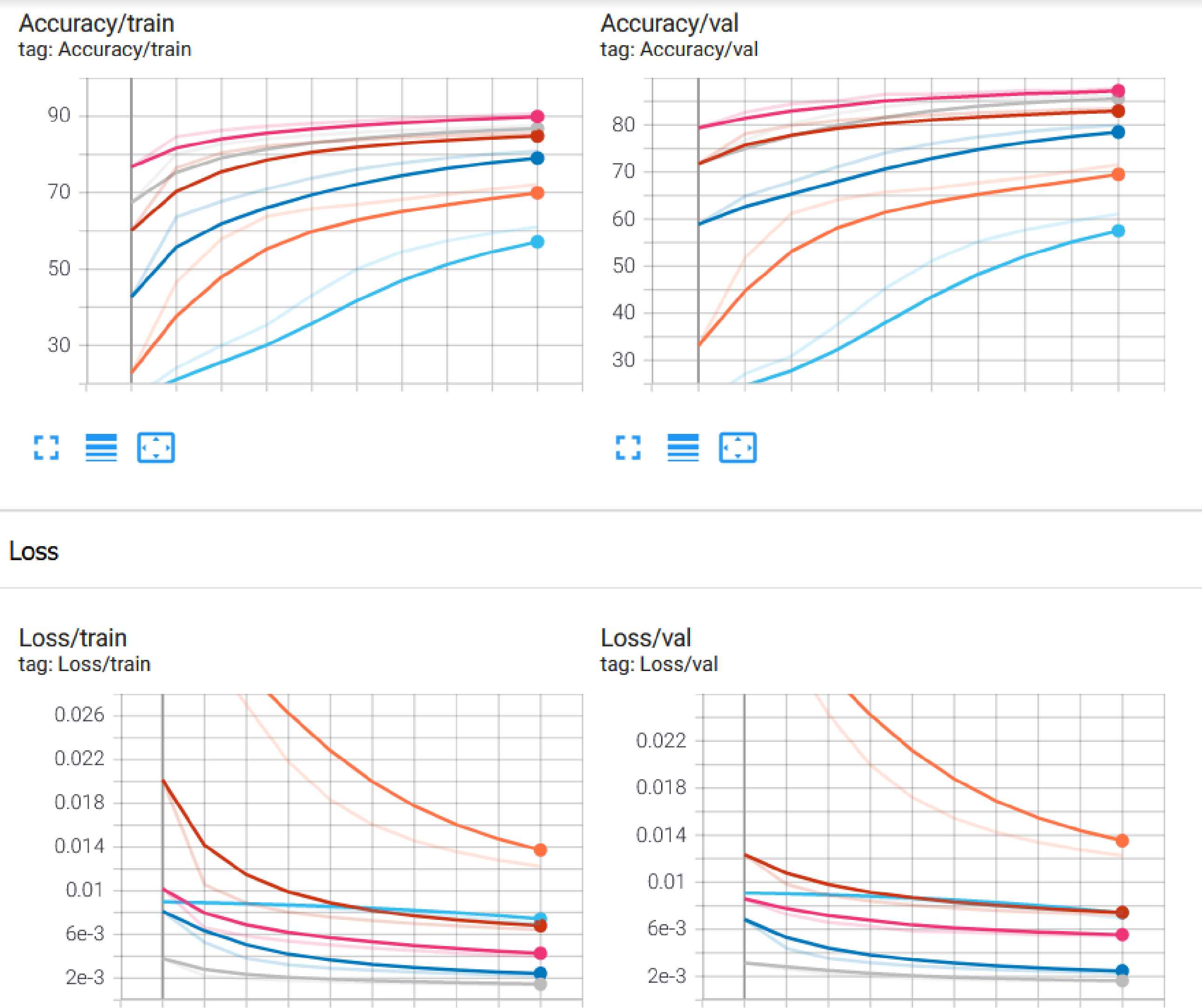


Runs

Write a regex to filter runs

- ☑ ○ runs/Jun13_16-48-43_16da605f563c
- ☑ ○ runs/Jun13_16-48-43_16da605f563c/1623602980.9918222
- ☑ ○ runs/Jun13_17-26-20_16da605f563c
- ☑ ○ runs/Jun13_17-26-20_16da605f563c/1623605237.2393396
- ☑ ○ runs/Jun13_17-28-20_16da605f563c
- ☑ ○ runs/Jun13_17-28-20_16da605f563c/1623605356.016401
- ☑ ○ runs/Jun13_17-32-57_16da605f563c
- runs/Jun13_17-32-57_16da605f563c/1

TOGGLE ALL RUNS



Accuracy/train
tag: Accuracy/train

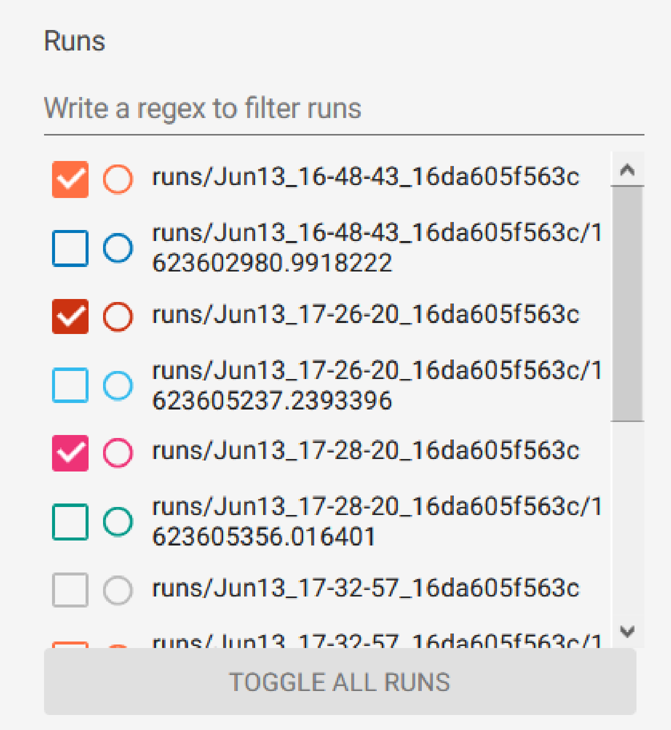
Accuracy/val
tag: Accuracy/val

Loss

Loss/train
tag: Loss/train

Loss/val
tag: Loss/val

TensorBoard Summary Writer



Runs

Write a regex to filter runs

☑ ○ runs/Jun13_16-48-43_16da605f563c

□ ○ runs/Jun13_16-48-43_16da605f563c/1623602980.9918222

☑ ○ runs/Jun13_17-26-20_16da605f563c

□ ○ runs/Jun13_17-26-20_16da605f563c/1623605237.2393396

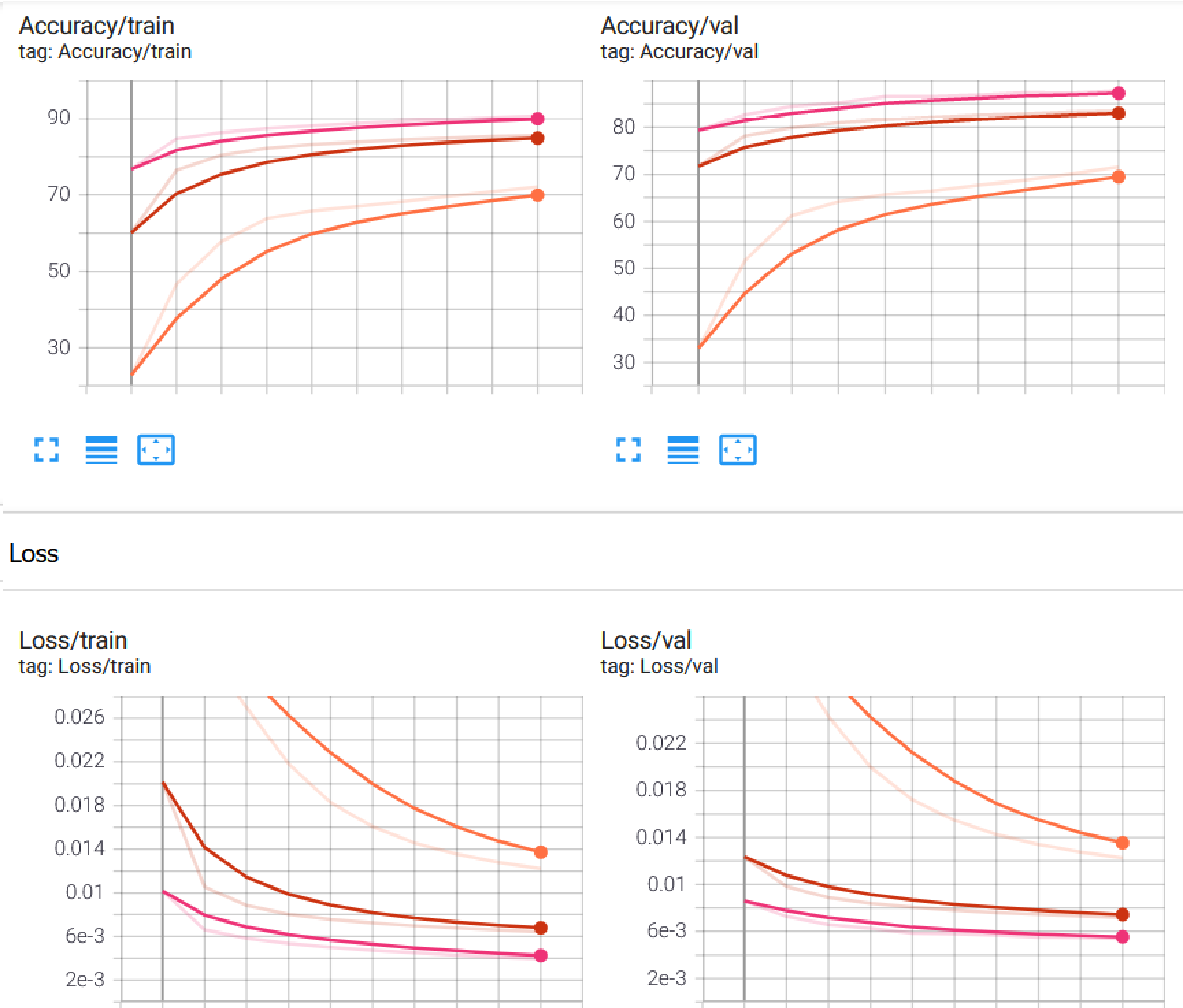
☑ ○ runs/Jun13_17-28-20_16da605f563c

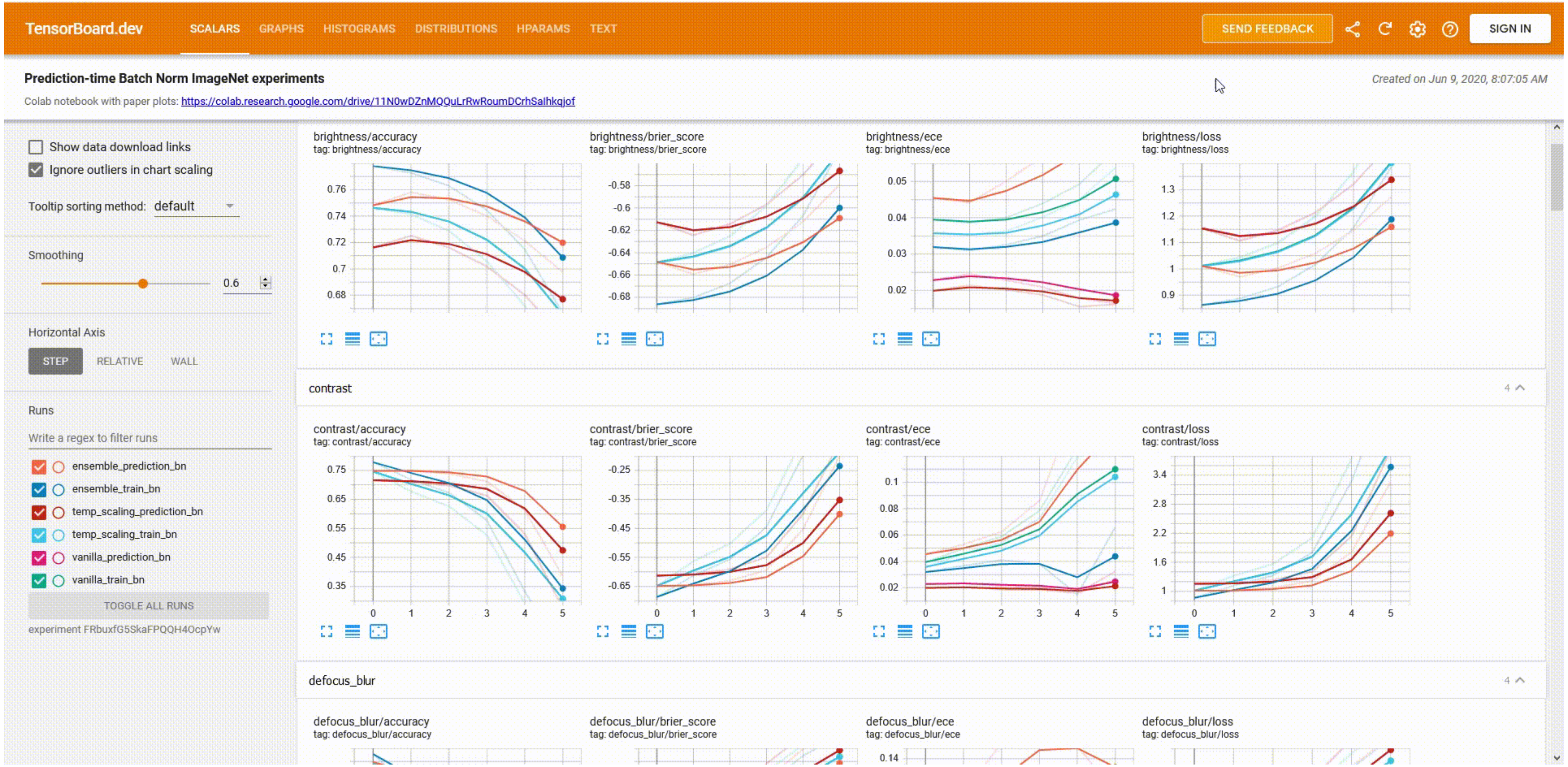
□ ○ runs/Jun13_17-28-20_16da605f563c/1623605356.016401

□ ○ runs/Jun13_17-32-57_16da605f563c

runs/Jun13_17-32-57_16da605f563c/1

TOGGLE ALL RUNS





Prediction-time Batch Norm ImageNet experiments

Colab notebook with paper plots: https://colab.research.google.com/drive/11N0wDZnMQQuLrRwRoumDCrhSalhkqjof

Created on Jun 9, 2020, 8:07:05 AM

☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

- ☑ ○ ensemble_prediction_bn
- ☑ ○ ensemble_train_bn
- ☑ ○ temp_scaling_prediction_bn
- ☑ ○ temp_scaling_train_bn
- ☑ ○ vanilla_prediction_bn
- ☑ ○ vanilla_train_bn

TOGGLE ALL RUNS

experiment FRbuxfG5SkaFPQQH4OcpYw

brightness/accuracy
tag: brightness/accuracy

brightness/brier_score
tag: brightness/brier_score

brightness/ece
tag: brightness/ece

brightness/loss
tag: brightness/loss

contrast

contrast/accuracy
tag: contrast/accuracy

contrast/brier_score
tag: contrast/brier_score

contrast/ece
tag: contrast/ece

contrast/loss
tag: contrast/loss

defocus_blur

defocus_blur/accuracy
tag: defocus_blur/accuracy

defocus_blur/brier_score
tag: defocus_blur/brier_score

defocus_blur/ece
tag: defocus_blur/ece

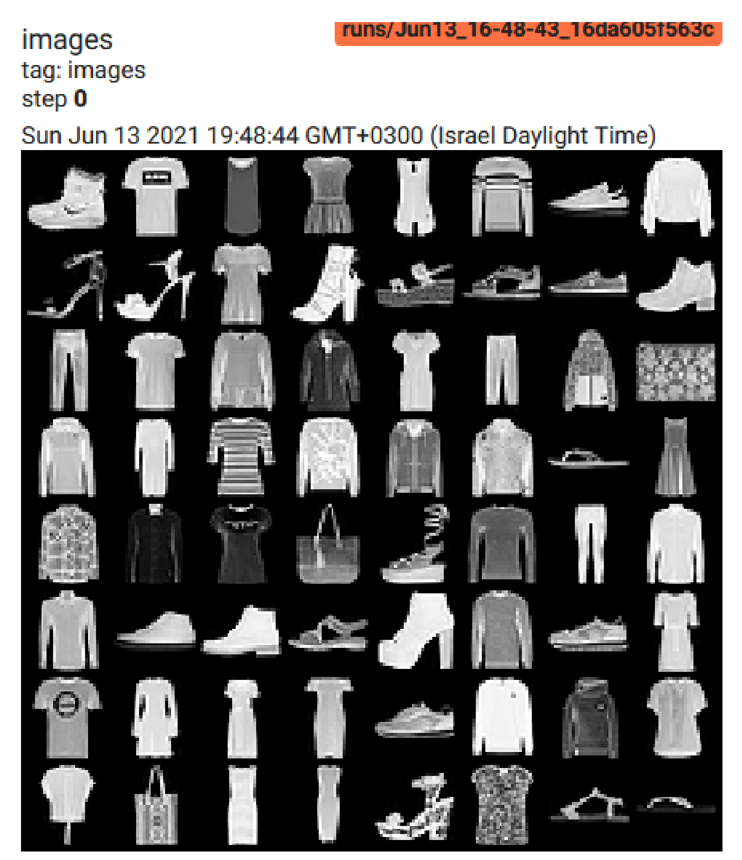
defocus_blur/loss
tag: defocus_blur/loss

Image credit to Tensorboard.dev

[redacted]

TensorBoard Log Images

```
# fetch a batch of data
X, y = next(iter(train_dataloader))

# create an image from all the images
grid = torchvision.utils.make_grid(X)

# add to summary
writer.add_image("images", grid)
```



images

runs/Jun13_16-48-43_16da605f563c

tag: images

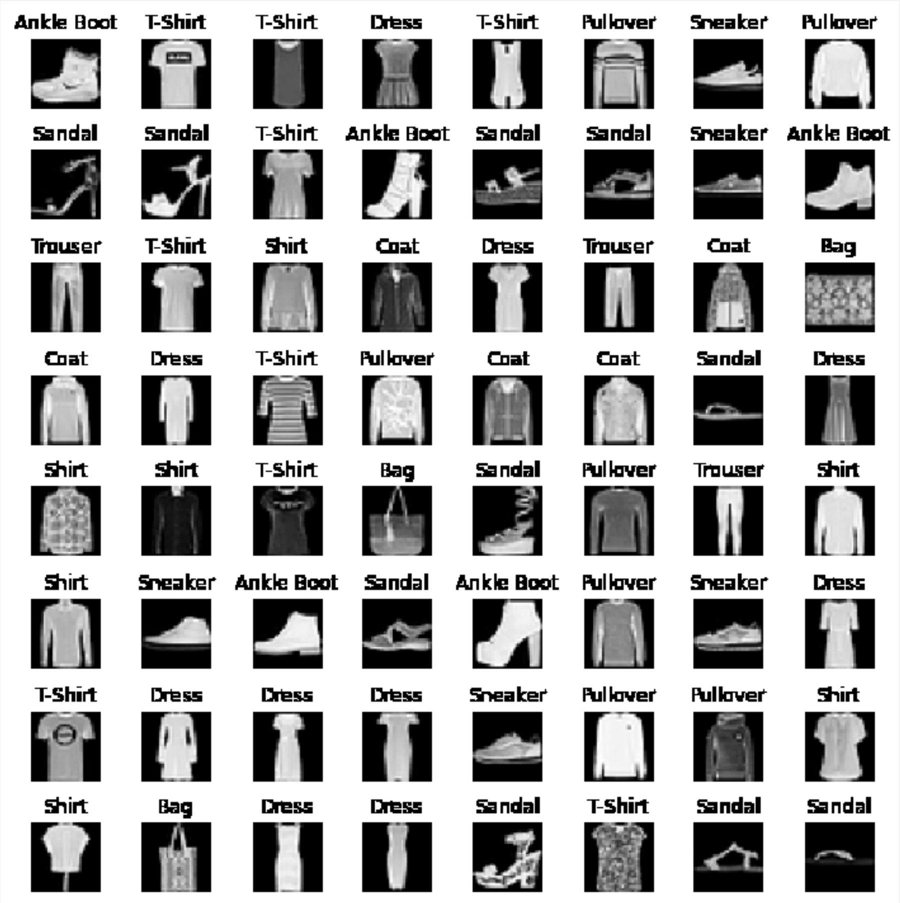
step **0**

Sun Jun 13 2021 19:48:44 GMT+0300 (Israel Daylight Time)

[redacted]

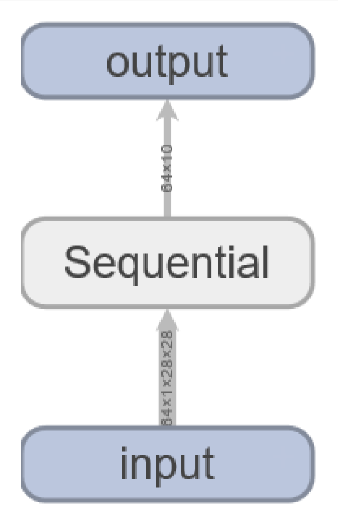
TensorBoard Log Figures

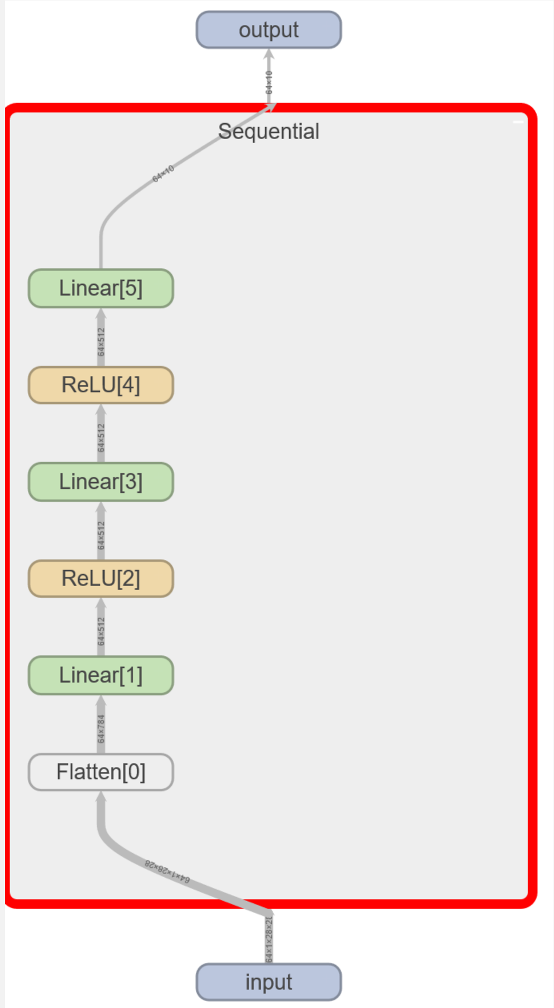
```
# create matplotlib figure
figure = ...

# add to summary
writer.add_figure("batch_example", figure)
```

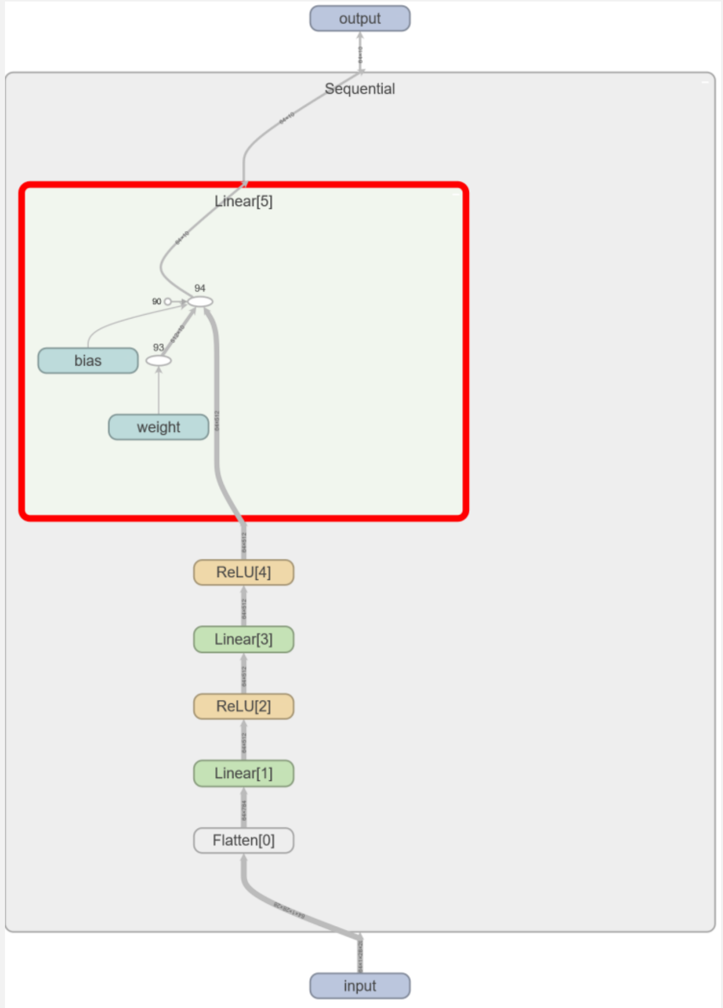


TensorBoard Graphs

```
writer.add_graph(model, X)
```



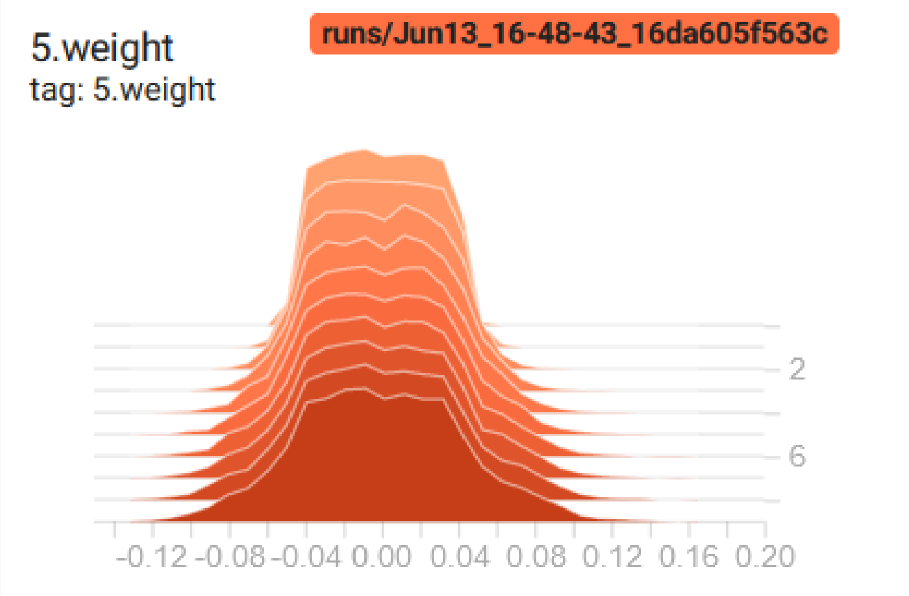


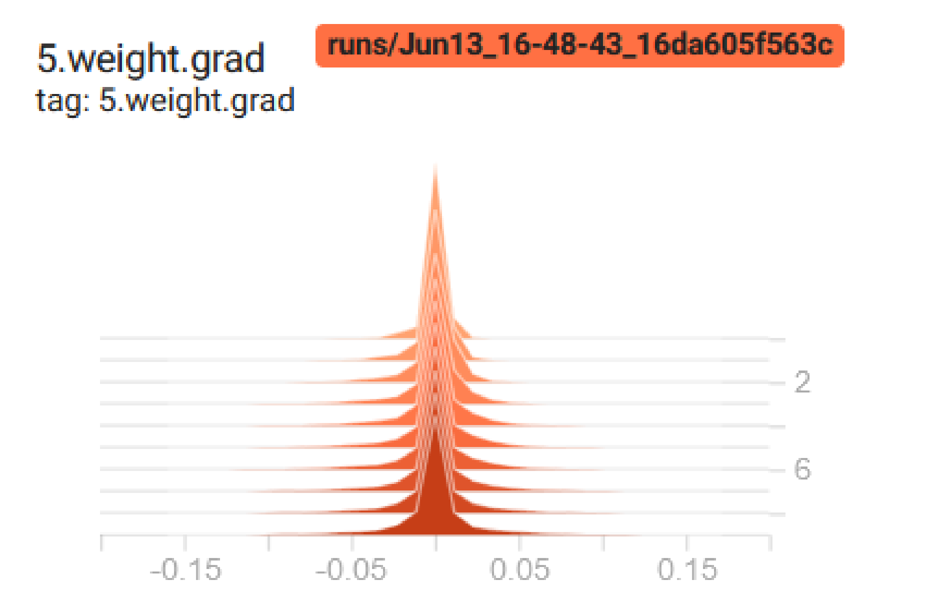


[redacted]

TensorBoard Histograms

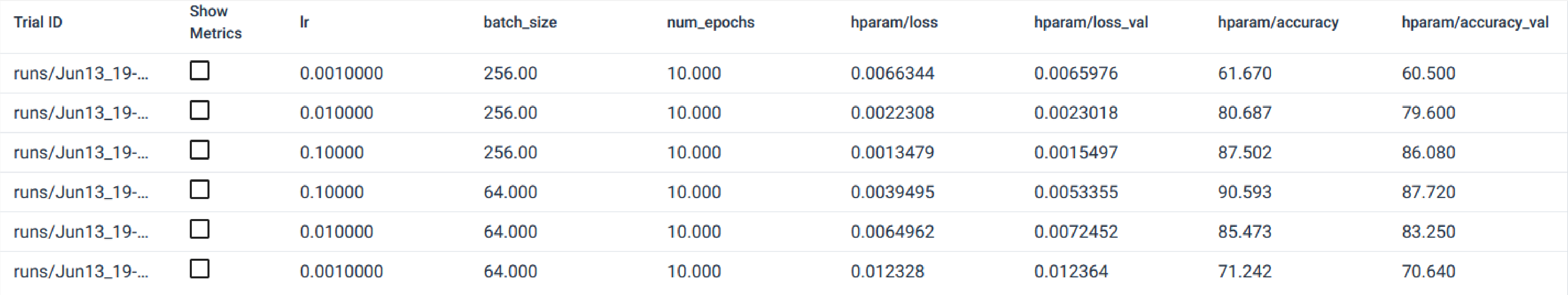
```
for name, weight in model.named_parameters():
  writer.add_histogram(name, weight, epoch)
  writer.add_histogram(f'{name}.grad', weight.grad, epoch)
```





TensorBoard Hyper-parameters

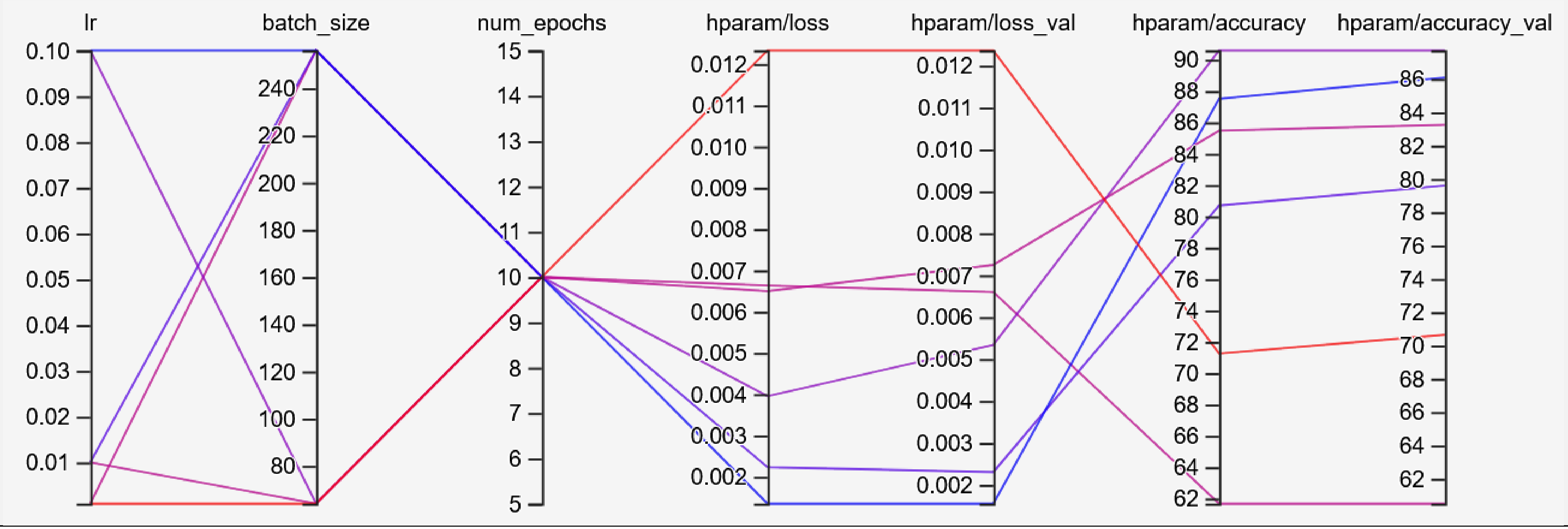
```
writer.add_hparams({'lr': LR, 'batch_size': B, 'num_epochs': EPOCHS},
                   {'hparam/loss': train_loss, 'hparam/loss_val': val_loss,
                    'hparam/accuracy': train_acc, 'hparam/accuracy_val’: val_acc})
```



Trial ID	Show Metrics	lr	batch_size	num_epochs	hparam/loss	hparam/loss_val	hparam/accuracy	hparam/accuracy_val
runs/Jun13_19-...	□	0.0010000	256.00	10.000	0.0066344	0.0065976	61.670	60.500
runs/Jun13_19-...	□	0.010000	256.00	10.000	0.0022308	0.0023018	80.687	79.600
runs/Jun13_19-...	□	0.10000	256.00	10.000	0.0013479	0.0015497	87.502	86.080
runs/Jun13_19-...	□	0.10000	64.000	10.000	0.0039495	0.0053355	90.593	87.720
runs/Jun13_19-...	□	0.010000	64.000	10.000	0.0064962	0.0072452	85.473	83.250
runs/Jun13_19-...	□	0.0010000	64.000	10.000	0.012328	0.012364	71.242	70.640

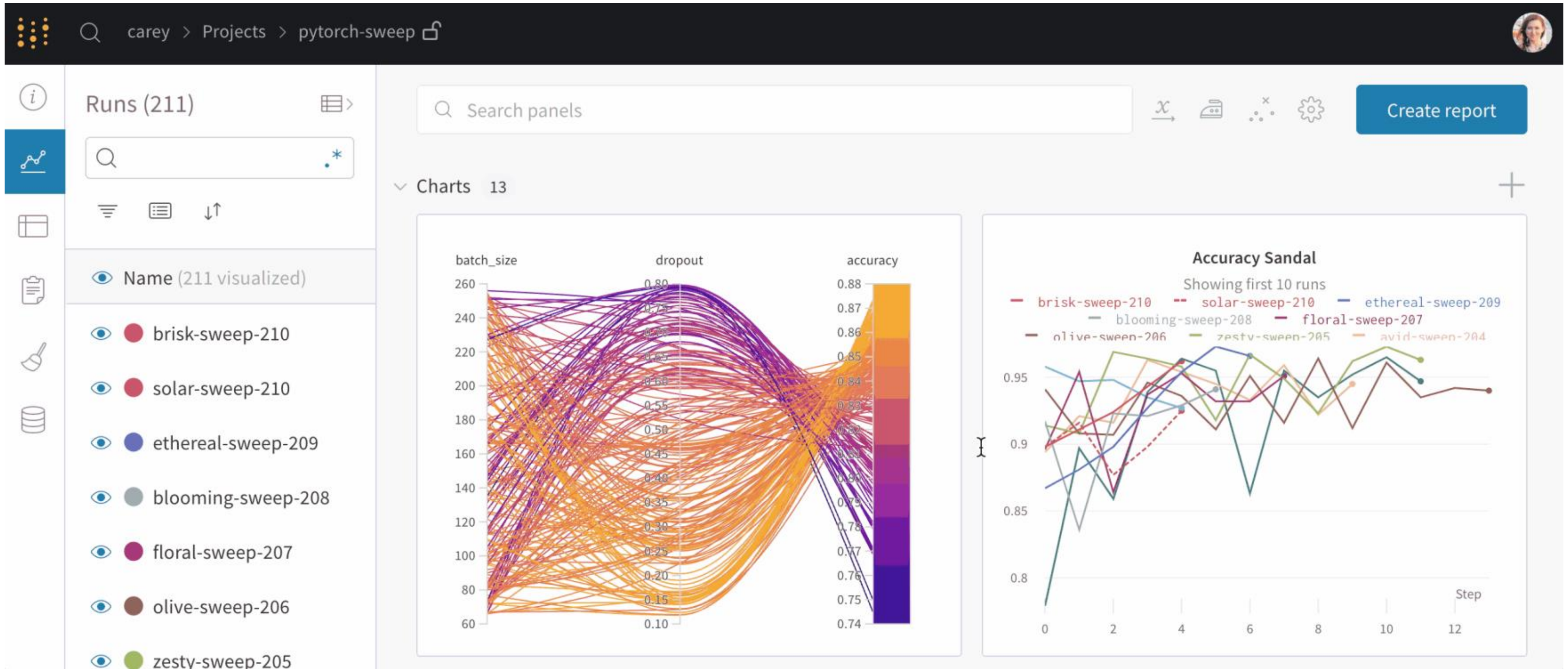
TensorBoard Hyper-parameters

```
writer.add_hparams({'lr': LR, 'batch_size': B, 'num_epochs': EPOCHS},
                   {'hparam/loss': train_loss, 'hparam/loss_val': val_loss,
                    'hparam/accuracy': train_acc, 'hparam/accuracy_val': val_loss})
```



Weights & Biases

- More features (sweeps, external access, working in groups, etc.)
- Requires registration, limited memory



Runs (211)

Name (211 visualized)

- brisk-sweep-210
- solar-sweep-210
- ethereal-sweep-209
- blooming-sweep-208
- floral-sweep-207
- olive-sweep-206
- zesty-sweep-205

[redacted]

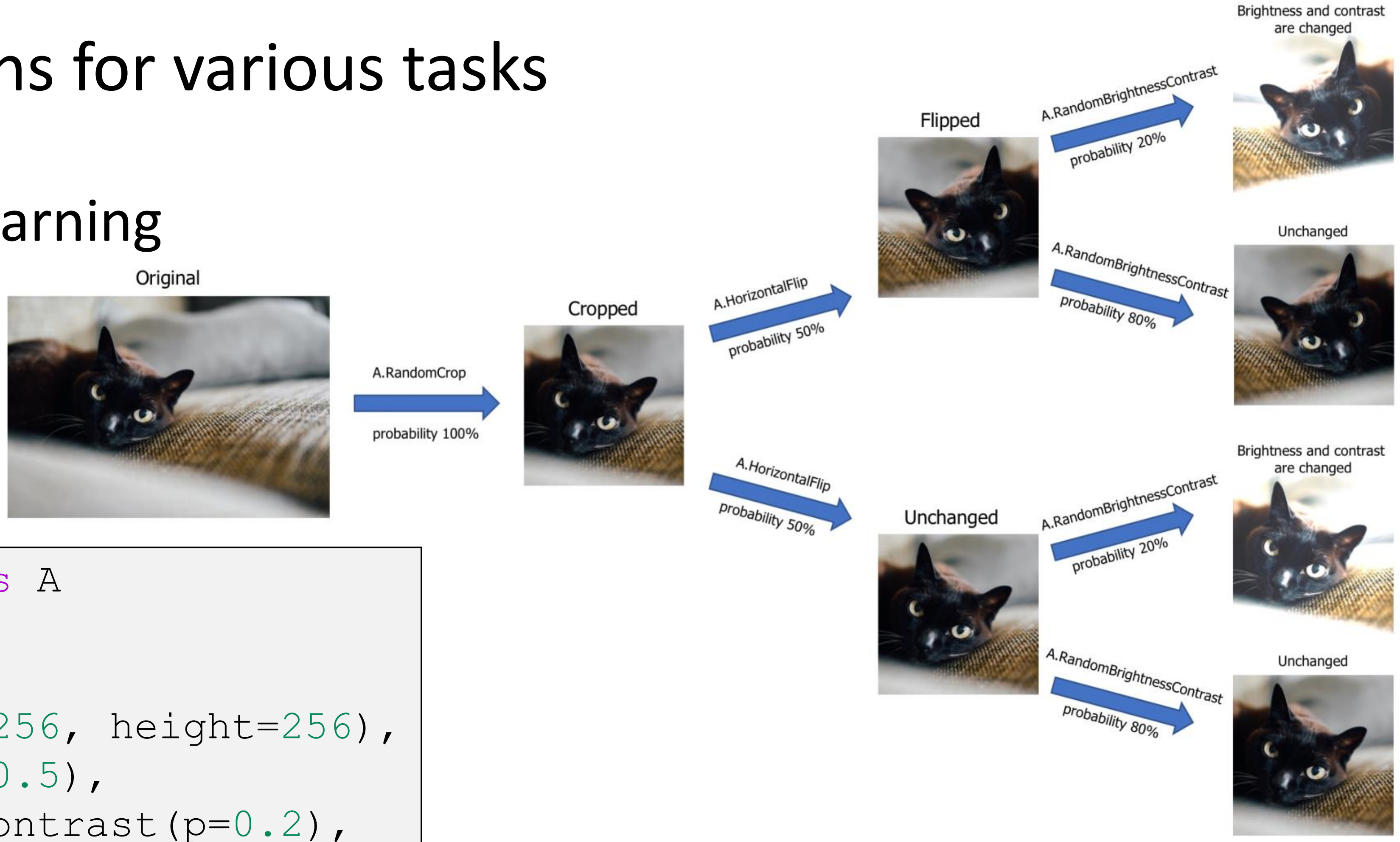
Don’t Reinvent the Wheel!

Use Existing Tools!

[redacted]

Albumentations

- Data Augmentations for various tasks
  - Classification / Representation Learning



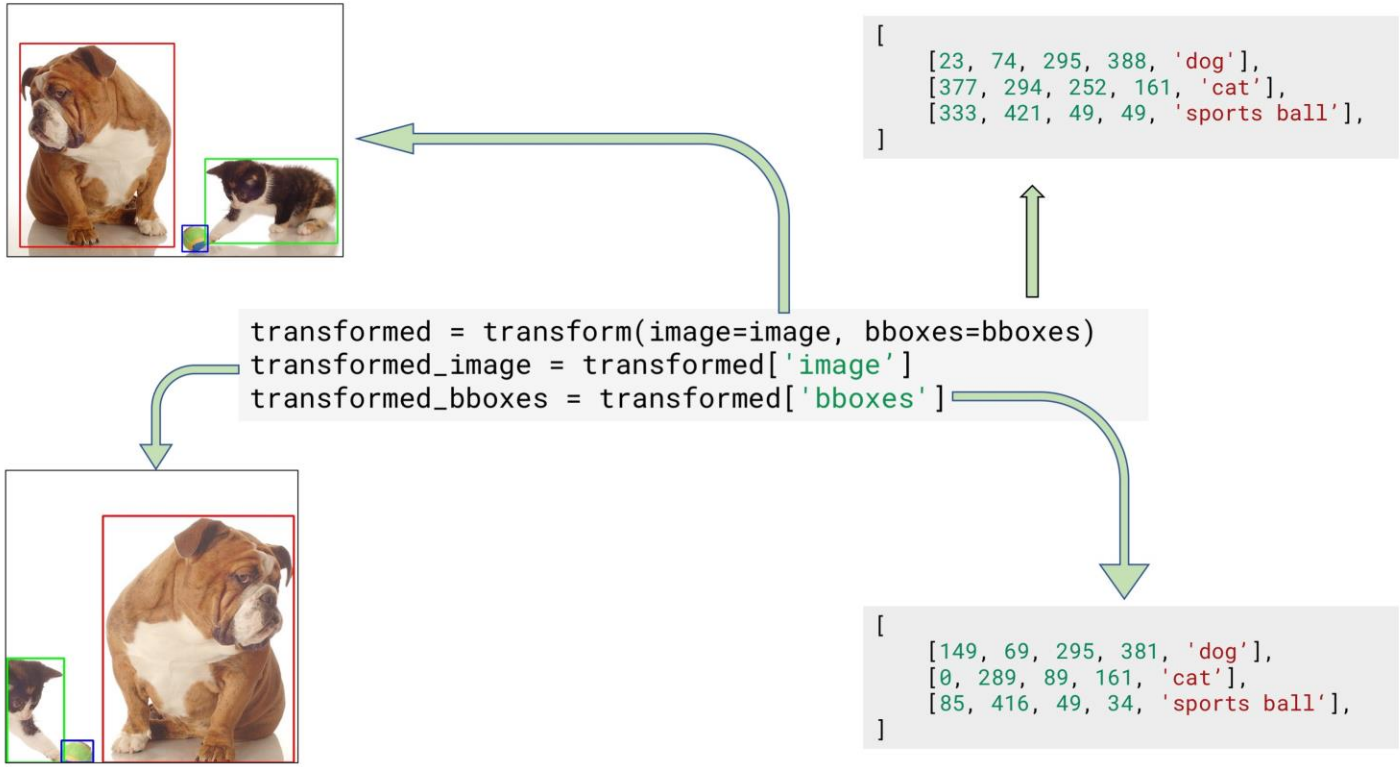
```
import albumentations as A

transform = A.Compose([
    A.RandomCrop(width=256, height=256),
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.2),
])
```

Image credit to albumentations tutorial:
https://albumentations.ai/docs/getting_started/image_augmentation/

[redacted]

Albumentations

- Data Augmentations for various tasks
  - Classification / Representation Learning
  - Object Detection



```
[
    [23, 74, 295, 388, 'dog'],
    [377, 294, 252, 161, 'cat'],
    [333, 421, 49, 49, 'sports ball'],
]
```

```
transformed = transform(image=image, bboxes=bboxes)
transformed_image = transformed['image']
transformed_bboxes = transformed['bboxes']
```

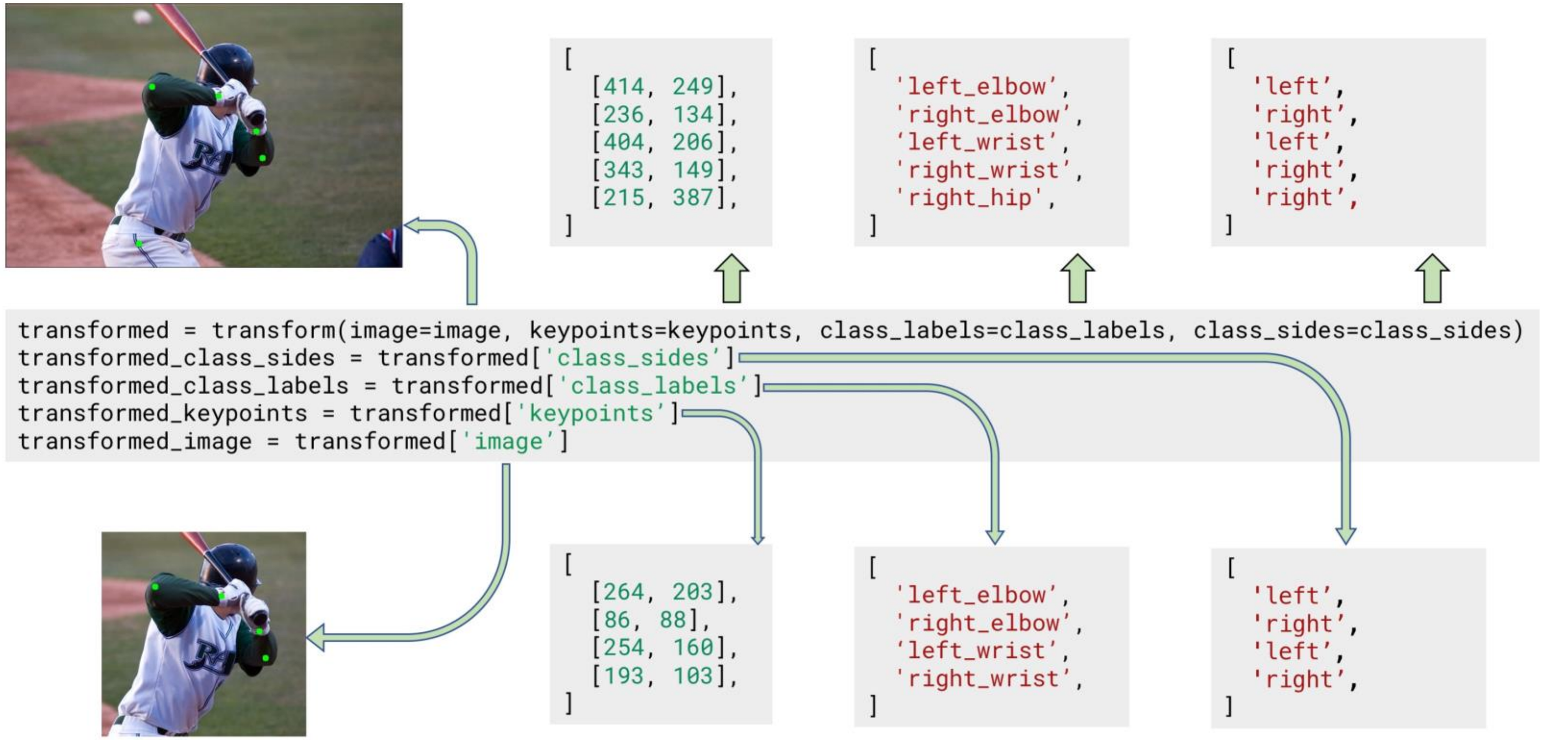
```
[
    [149, 69, 295, 381, 'dog'],
    [0, 289, 89, 161, 'cat'],
    [85, 416, 49, 34, 'sports ball'],
]
```

Image credit to albumentations tutorial: https://albumentations.ai/docs/getting_started/bounding_boxes_augmentation/

[redacted]

Albumentations

- Data Augmentations for various tasks
  - Classification / Representation Learning
  - Object Detection
  - Keypoint Detection



```
[
  [414, 249],
  [236, 134],
  [404, 206],
  [343, 149],
  [215, 387],
]
```

```
[
  'left_elbow',
  'right_elbow',
  'left_wrist',
  'right_wrist',
  'right_hip',
]
```

```
[
  'left',
  'right',
  'left',
  'right',
  'right',
]
```

```
transformed = transform(image=image, keypoints=keypoints, class_labels=class_labels, class_sides=class_sides)
transformed_class_sides = transformed['class_sides']
transformed_class_labels = transformed['class_labels']
transformed_keypoints = transformed['keypoints']
transformed_image = transformed['image']
```

```
[
  [264, 203],
  [86, 88],
  [254, 160],
  [193, 103],
]
```

```
[
  'left_elbow',
  'right_elbow',
  'left_wrist',
  'right_wrist',
]
```

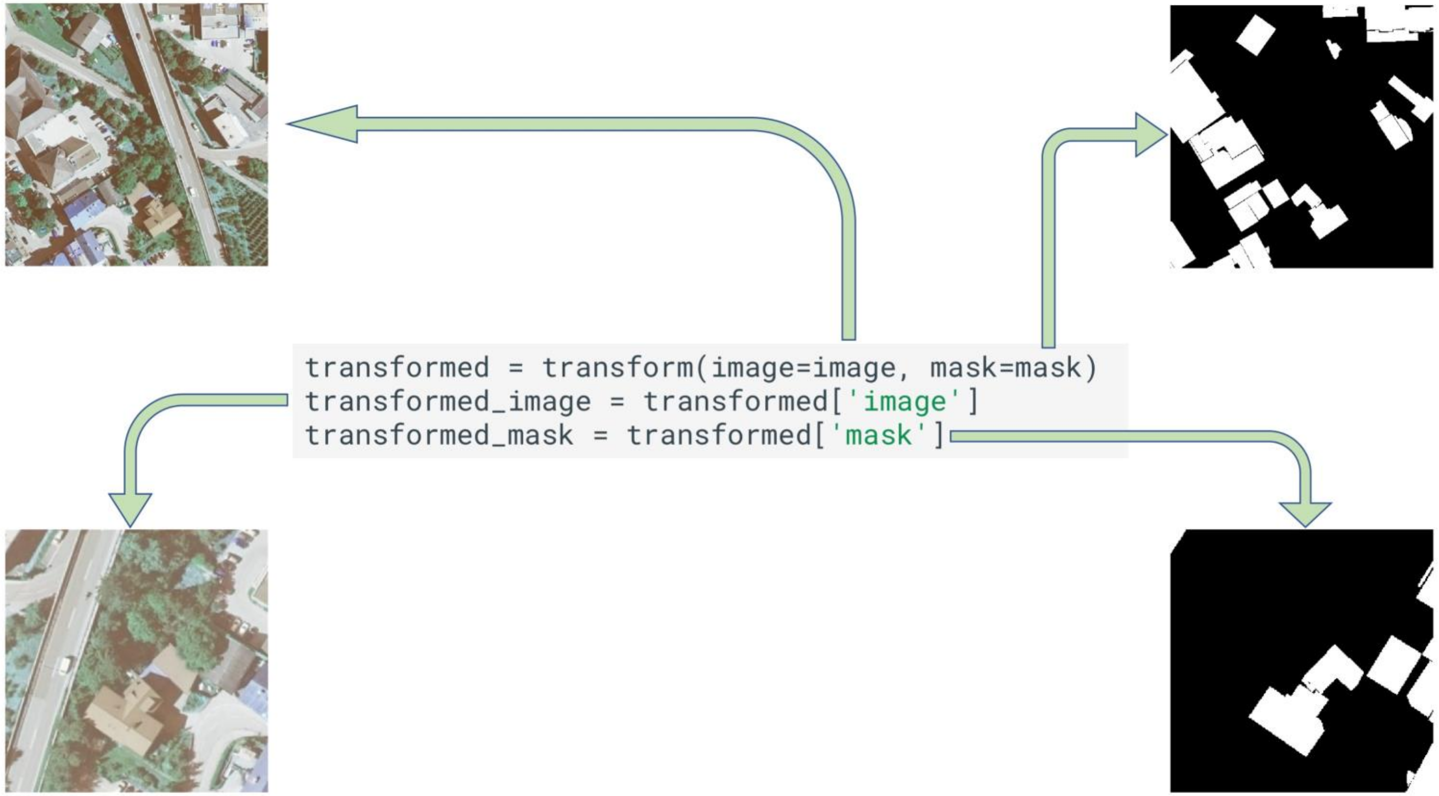
```
[
  'left',
  'right',
  'left',
  'right',
]
```

Image credit to albumentations tutorial:
https://albumentations.ai/docs/getting_started/keypoints_augmentation/

[redacted]

Albumentations

- Data Augmentations for various tasks
  - Classification / Representation Learning
  - Object Detection
  - Keypoint Detection
  - Mask Segmentation

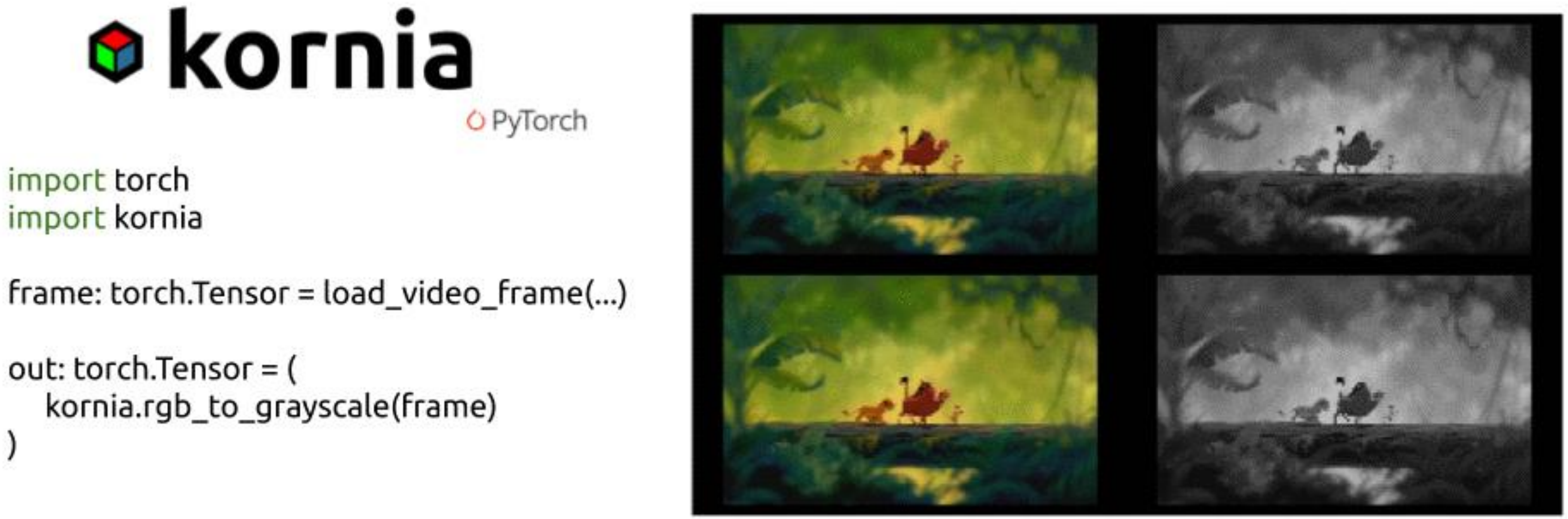


```
transformed = transform(image=image, mask=mask)
transformed_image = transformed['image']
transformed_mask = transformed['mask']
```

Image credit to albumentations tutorial:
https://albumentations.ai/docs/getting_started/mask_augmentation/

[redacted]

kornia



```
import torch
import kornia

frame: torch.Tensor = load_video_frame(...)

out: torch.Tensor = (
    kornia.rgb_to_grayscale(frame)
)
```

FULLY DIFFERENTIABLE

kornia

```
# compute perspective transform
M = K.get_perspective_transform(points_src, points_dst)

# warp the original image by the found transform
img_warp = K.warp_perspective(img.float(), M, dsize=(h, w))
```

image source

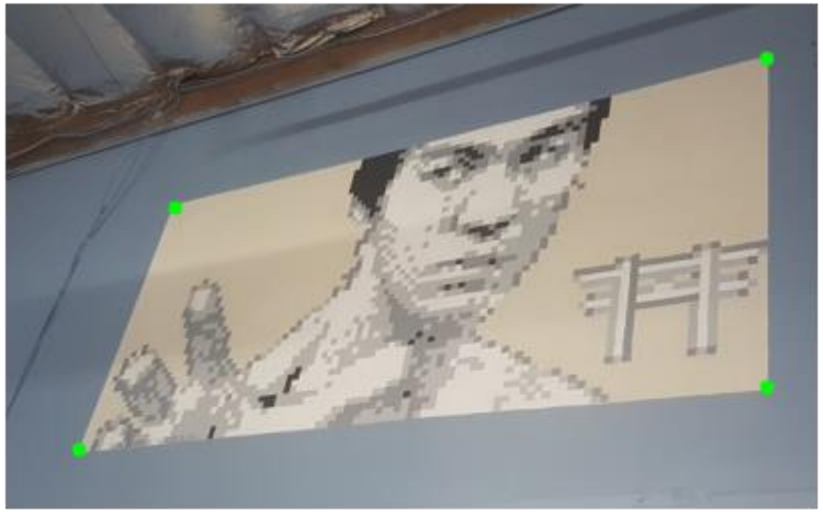


image destination

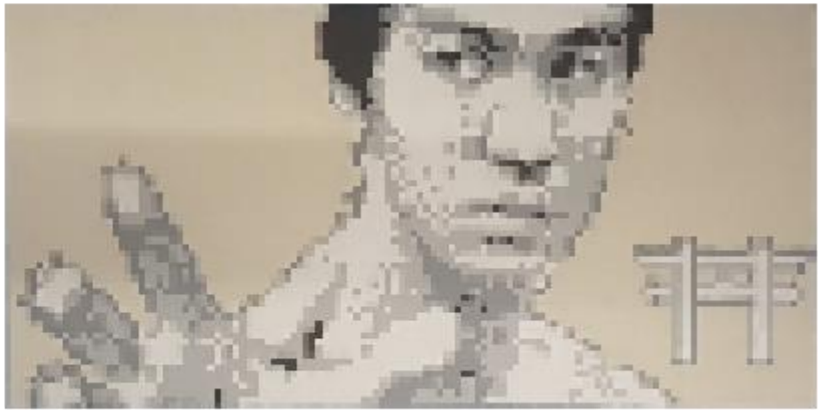


Image credit to kornia tutorial:
https://kornia-tutorials.readthedocs.io/en/latest/warp_perspective.html

kornia

```
# create the operator
canny = K.filters.Canny()

# blur the image
x_magnitude, x_canny = canny(data.float())
```

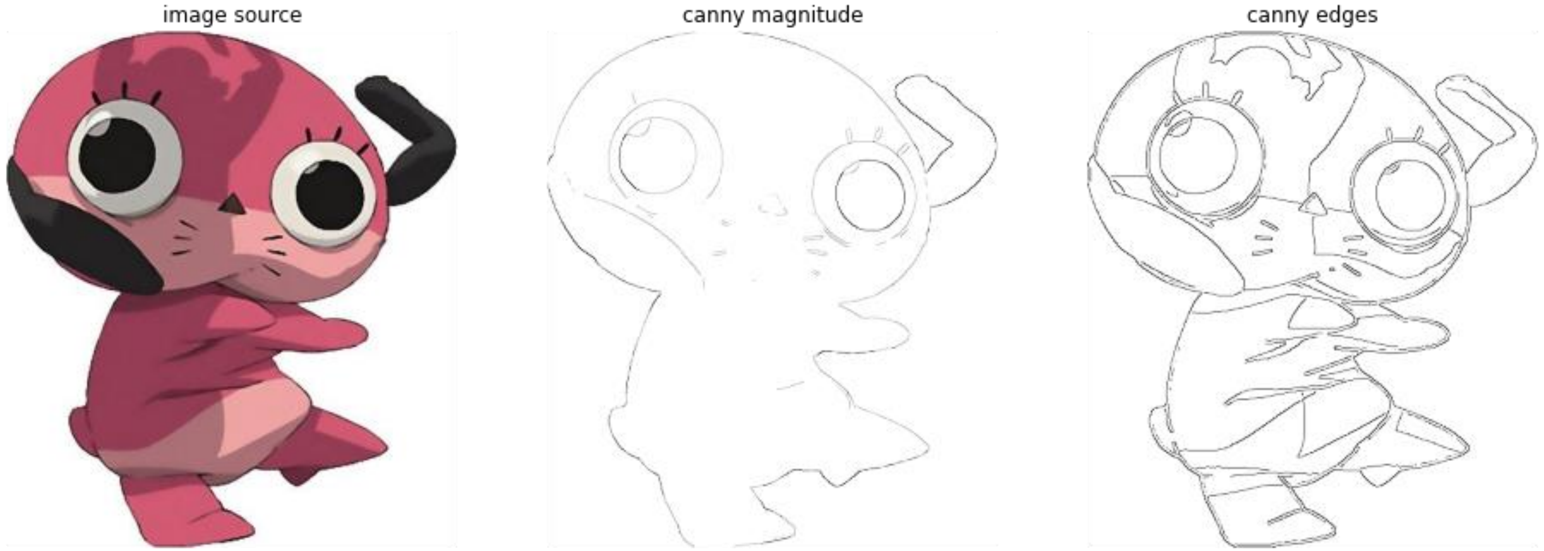


image source

canny magnitude

canny edges

Image credit to kornia tutorial:
https://kornia-tutorials.readthedocs.io/en/latest/canny.html

[redacted]

kornia

```
# create the operator
gauss = K.filters.GaussianBlur2d((11, 11), (10.5, 10.5))

# blur the image
x_blur = gauss(data.float())
```

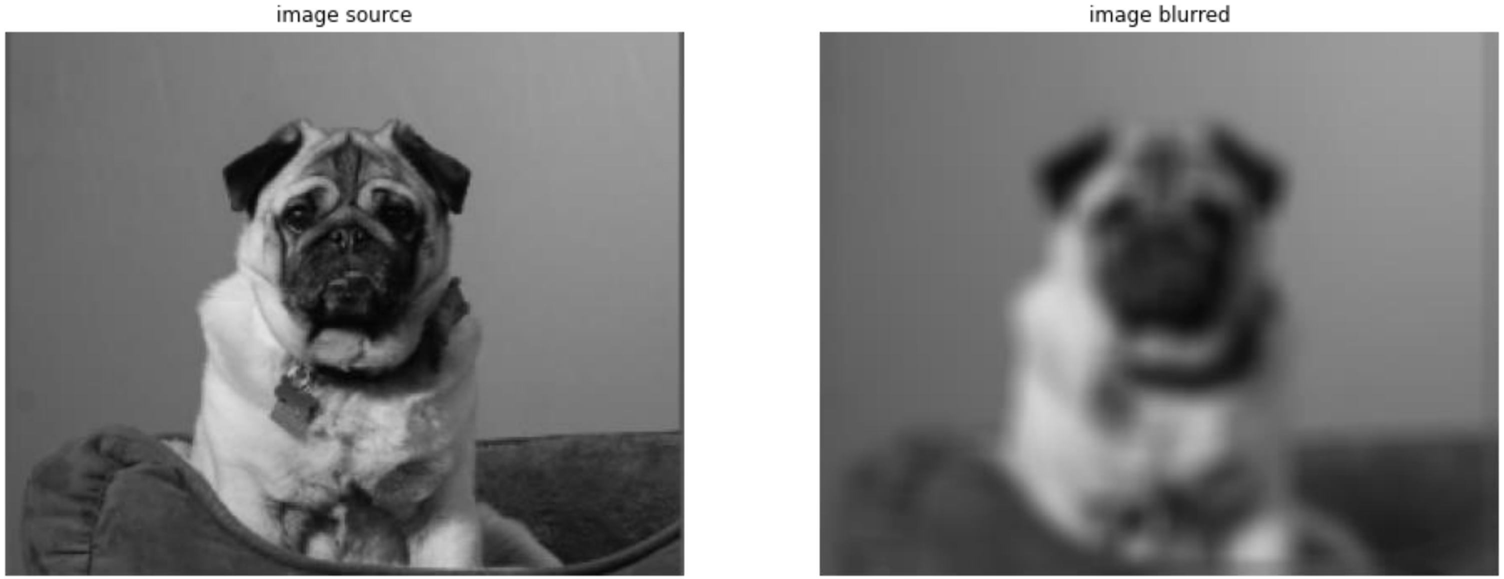


Image credit to kornia tutorial:
https://kornia-tutorials.readthedocs.io/en/latest/gaussian_blur.html

[redacted]

kornia

```
# define sharpening mask
sharpen = kornia.filters.UnsharpMask((9,9), (2.5,2.5))

# create the sharpened image
sharpened_tensor = sharpen(data)

# get difference between original and sharpened image
difference = (sharpened_tensor - data).abs()
```

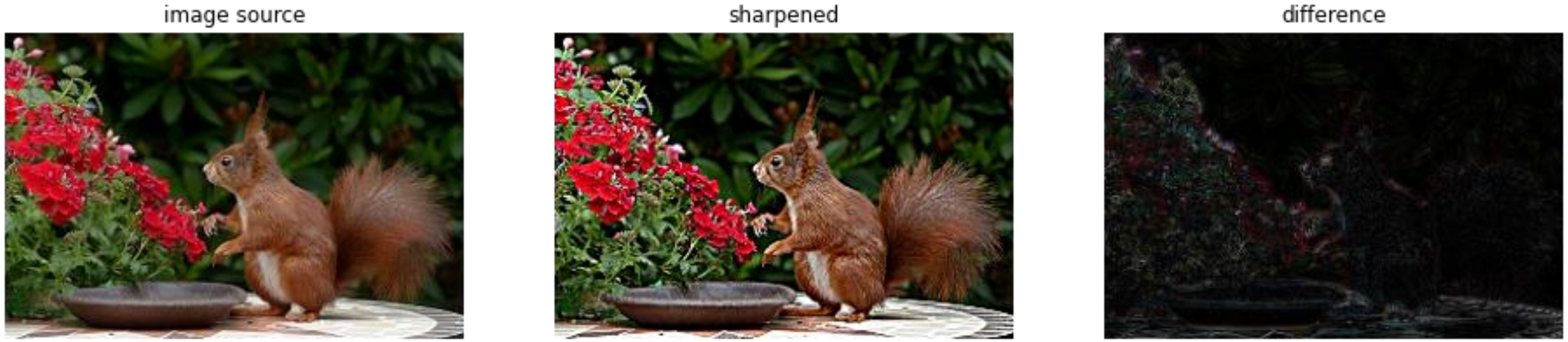


image source

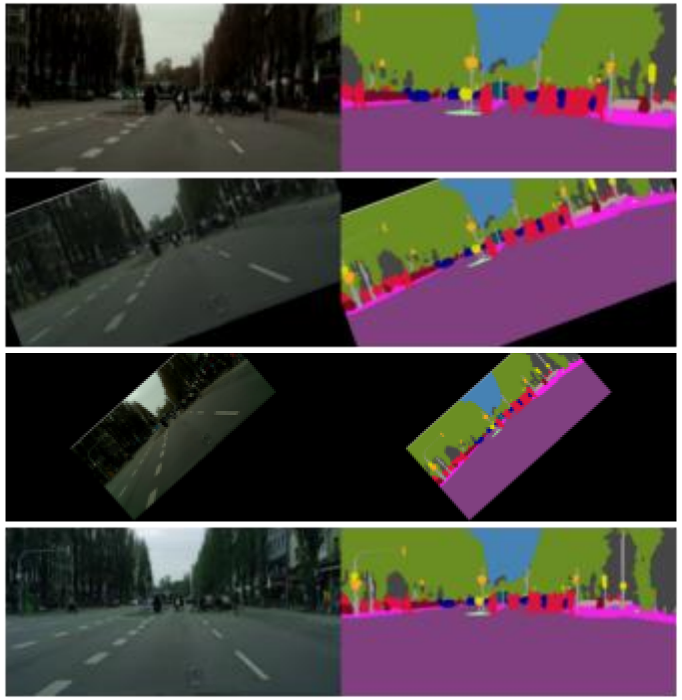
sharpened

difference

Image credit to kornia tutorial:
https://kornia-tutorials.readthedocs.io/en/latest/unsharp_mask.html

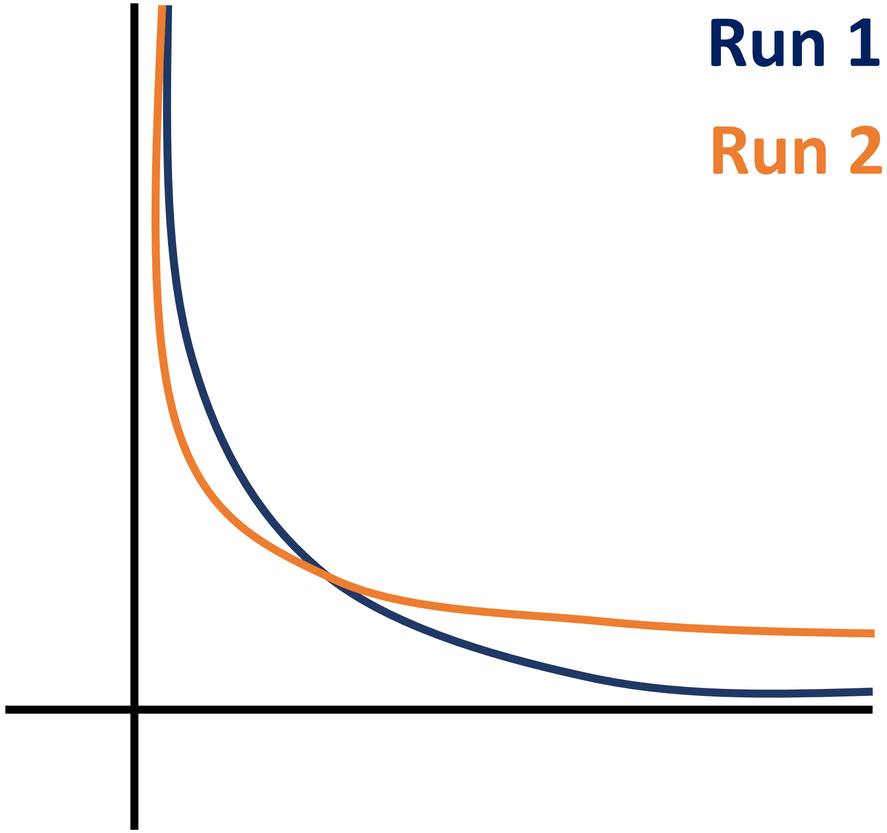
kornia

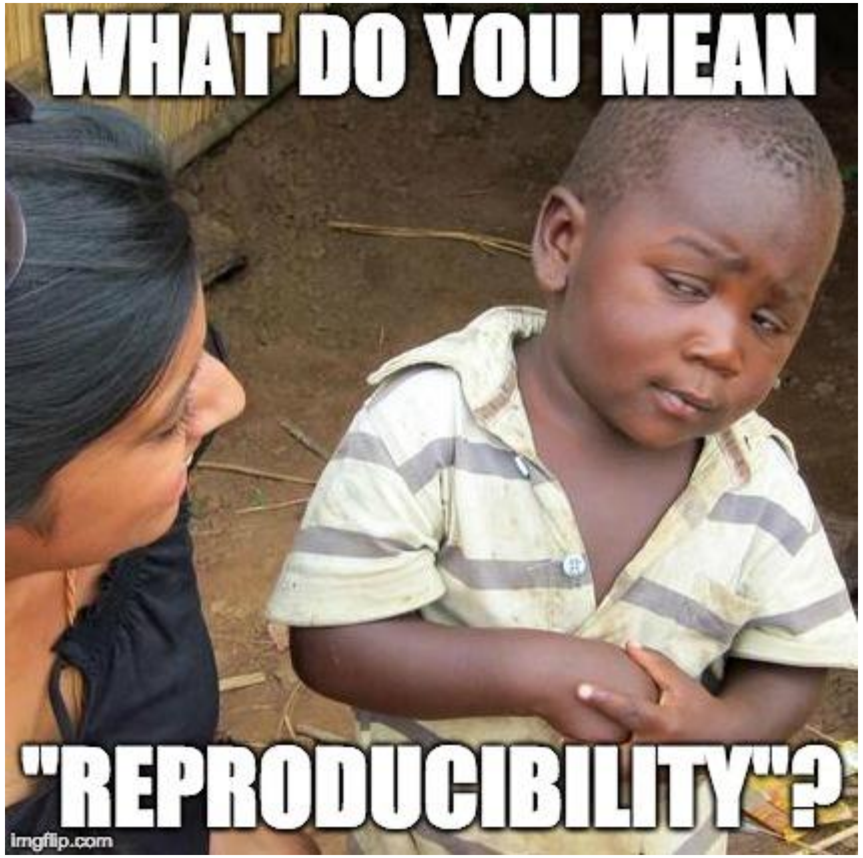
```
import torch
import torch.nn as nn
import kornia as K

img = load_image(...) #BxCxHxW

aug = nn.Sequential(
    K.augmentations.ColorJitter(0.15, 0.25, 0.25, 0.25),
    K.augmentation.RandomAffine([-45., 45.], [0., 0.15],
                                [0.5, 1.5], [0., 0.15]),
)

out = aug(img) #BxCxHxW
```



[redacted]

Reproducibility





[redacted]

Reproducibility

```
# set random seeds
seed = 42
torch.manual_seed(seed)
random.seed(seed)
np.random.seed(seed)

# use deterministic algorithms only
torch.use_deterministic_algorithms(True)

# use known convolution algorithm in
cudnn
torch.backends.cudnn.benchmark = False
```

```
# fix workers randomness

def seed_worker(worker_id):
    worker_seed = torch.initial_seed() %
2**32
    numpy.random.seed(worker_seed)
    random.seed(worker_seed)

g = torch.Generator()
g.manual_seed(seed)

DataLoader(
    train_dataset,
    batch_size=batch_size,
    num_workers=num_workers,
    worker_init_fn=seed_worker,
    generator=g
)
```

Base on:
https://pytorch.org/docs/stable/notes/randomness.html

[redacted]

Saving & Loading Models

Serialize entire model

```
torch.save(model, "my_model.pth")
...
model = torch.load("my_model.pth")
```

```
# save an object to disk
torch.save(object, path)

# load an object from disk
object = torch.load(path)
```

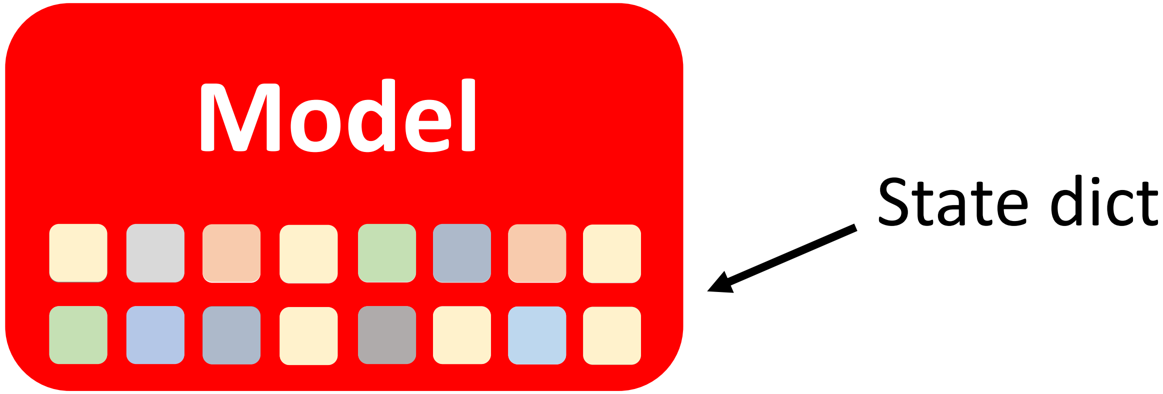
Pros:

Simple

Cons:

Can’t change the model class

Can’t continue training



Base on:
https://pytorch.org/tutorials/beginner/saving_loading_models.html

Saving & Loading Models

```python
# Define model
class TheModelClass(nn.Module):
    def __init__(self):
        super(TheModelClass, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        ...

# Initialize model
model = TheModelClass()
```

```
Model's state_dict:
conv1.weight torch.Size([6,3,5,5])
conv1.bias   torch.Size([6])
conv2.weight torch.Size([16,6,5,5])
conv2.bias   torch.Size([16])
fc1.weight   torch.Size([120, 400])
fc1.bias     torch.Size([120])
fc2.weight   torch.Size([84, 120])
fc2.bias     torch.Size([84])
fc3.weight   torch.Size([10, 84])
fc3.bias     torch.Size([10])
```

Base on:
https://pytorch.org/tutorials/beginner/saving_loading_models.html

[redacted]

Saving & Loading Models

Saving the better way

```
# save the model’s state dict
torch.save(model.state_dict(), "my_model.pth")

...

# create and load the model’s state dict
model = TheModelClass(*args, **kwargs)
model.load_state_dict(torch.load("my_model.pth"))
```

```
# save an object to disk
torch.save(object, path)

# load an object from disk
torch.load(path)

# load the state dict to a
model
model.load_state_dict(sd)
```

Base on:
https://pytorch.org/tutorials/beginner/saving_loading_models.html

[redacted]

Saving & Loading Models

```
# Initialize optimizer
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

```
Optimizer's state_dict:
state    {}
param_groups     [{'lr': 0.001, 'momentum': 0.9, 'weight_decay': 0, ...]
```

Base on:
https://pytorch.org/tutorials/beginner/saving_loading_models.html

[redacted]

Saving & Loading Models

Saving for training

```
checkpoint = torch.save({'epoch': epoch,
                         'model_sd': model.state_dict(),
                         'opt_sd': optimizer.state_dict(),
                         'loss': loss,
                         ...}, 'checkpoint.pth')
```

```
model = TheModelClass(*args, **kwargs)
optimizer = TheOptimizerClass(*args, **kwargs)

checkpoint = torch.load('checkpoint.pth')
model.load_state_dict(checkpoint['model_sd'])
optimizer.load_state_dict(checkpoint['opt_sd'])
epoch = checkpoint['epoch']
loss = checkpoint['loss’]
# continue training
```

Base on:
https://pytorch.org/tutorials/beginner/saving_loading_models.html

[redacted]

[redacted]

Next week:

Computer Graphics and Rendering

[redacted]

Sources

- Tensorboard https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html
- https://medium.com/@iamsdt/using-tensorboard-in-google-colab-with-pytorch-458f9bb95212
- https://towardsdatascience.com/a-complete-guide-to-using-tensorboard-with-pytorch-53cb2301e8c3