distortion_classifier

November 23, 2025

[1]:
```
import os
from tqdm import tqdm
import torch
import torchvision
import torch.nn as nn
import numpy as np
import torch.nn.functional as F
import torchvision.transforms as T
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
from torch.utils.data import random_split
from torchvision.utils import make_grid
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

matplotlib.rcParams['figure.facecolor'] = '#ffffff'
```

[2]:
```
ROOT_DIR = os.path.abspath(os.path.join(os.getcwd(), '..', '..'))
DATA_DIR = os.path.join(ROOT_DIR, 'data', 'cnn')
print(os.listdir(DATA_DIR))
classes = os.listdir(os.path.join(DATA_DIR, 'train')) # Must remove enhanced␣
 ↪folder
print(classes)
```

```
['test', 'test2', 'test3', 'train', 'valid']
['h_blur', 'low_light', 'low_qual', 'normal', 'v_blur']
```

0.0.1 Data Transformation

[4]:
```
# Data transforms (normalization & data augmentation)
stats = ((0.42720765, 0.43359186, 0.44090385), (0.24280364, 0.24590165, 0.
 ↪23916515))
train_tfms = T.Compose([T.Resize([128, 256]),
                        T.RandomPerspective(distortion_scale=0.2, p=0.2),
                        T.ColorJitter(hue=.2),
                        T.ToTensor(),
                        T.Normalize(*stats, inplace=True)])
```

```
valid_tfms = T.Compose([T.Resize([128, 256]), T.ToTensor(), T.
 ↪Normalize(*stats)])
```

0.0.2 PyTorch Datasets / Data Loaders

[6]:
```
train_ds = ImageFolder(os.path.join(DATA_DIR, 'train'), train_tfms)
valid_ds = ImageFolder(os.path.join(DATA_DIR, 'valid'), valid_tfms)
```

[7]:
```
batch_size = 64
```

[8]:
```
# PyTorch Data Loaders
train_dl = DataLoader(train_ds, batch_size, shuffle=True, num_workers=3,␣
 ↪pin_memory=True)
valid_dl = DataLoader(valid_ds, batch_size*2, num_workers=3, pin_memory=True)
```

[9]:
```
def denormalize(images, means, stds):
    means = torch.tensor(means).reshape(1, 3, 1, 1)
    stds = torch.tensor(stds).reshape(1, 3, 1, 1)
    return images * stds + means

def show_batch(dl):
    for images, labels in dl:
        fig, ax = plt.subplots(figsize=(12, 12))
        ax.set_xticks([]); ax.set_yticks([])
        denorm_images = denormalize(images, *stats)
        ax.imshow(make_grid(denorm_images[:32], nrow=8).permute(1, 2, 0).
 ↪clamp(0, 1))
        break
```

[35]:
```
show_batch(train_dl)
```

0.0.3 Using GPU

[10]:
```
def get_default_device():
    if torch.cuda.is_available():
        return torch.device('cuda')
    return torch.device('cpu')

def to_device(data, device):
    if isinstance(data, (list, tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():
    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):
        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        return len(self.dl)
```

[11]:
```
device = get_default_device()
device
```

[11]:
```
device(type='cuda')
```

[12]:
```
train_dl = DeviceDataLoader(train_dl, device)
valid_dl = DeviceDataLoader(valid_dl, device)
```

0.0.4 Model Development

[14]:
```
def accuracy(outputs, labels):
    _, preds = torch.max(outputs, dim=1)
    return torch.tensor(torch.sum(preds == labels).item() / len(preds))

class ImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, labels = batch
        out = self(images)
        loss = F.cross_entropy(out, labels)
        return loss

    def validation_step(self, batch):
        images, labels = batch
        out = self(images)
```

```
        loss = F.cross_entropy(out, labels)
        acc = accuracy(out, labels)
        return {'val_loss': loss.detach(), 'val_acc': acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()        # Combine losses
        batch_accs = [x['val_acc'] for x in outputs]
        epoch_acc = torch.stack(batch_accs).mean()           # Combine␣
↪accuracies
        return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

    def epoch_end(self, epoch, result):
        print(f'Epoch [{epoch}], last_lr: {result['lrs'][-1]:.5f}, train_loss:␣
↪{result['train_loss']:.4f},' +
              f'val_loss: {result['val_loss']:.4f}, val_acc: {result['val_acc']:
↪.4f}')
```

[15]:
```
def conv_block(in_channels, out_channels, pool=False):
    layers = [nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1),
              nn.BatchNorm2d(out_channels),
              nn.ReLU(inplace=True)]
    if pool: layers.append(nn.MaxPool2d(2))
    return nn.Sequential(*layers)

class DistortionClassifier(ImageClassificationBase):
    def __init__(self, in_channels, num_classes):
        super().__init__()

        # 3 x 128 x 256
        self.conv1 = conv_block(in_channels, 64, pool=True) # 64 x 64 x 128
        self.conv2 = conv_block(64, 128, pool=True) # 128 x 32 x 64
        self.res1 = nn.Sequential(conv_block(128, 128), conv_block(128, 128)) #␣
↪128 x 32 x 64

        self.conv3 = conv_block(128, 256, pool=True) # 256 x 16 x 32
        self.conv4 = conv_block(256, 512, pool=True) # 512 x 8 x 16
        self.res2 = nn.Sequential(conv_block(512, 512), conv_block(512, 512)) #␣
↪512 x 8 x 16

        self.classifier = nn.Sequential(nn.MaxPool2d(2), # 512 x 4 x 8
                                        nn.Flatten(),
                                        nn.Dropout(0.2),
                                        nn.Linear(512 * 4 * 8, num_classes))

    def forward(self, xb):
        out = self.conv1(xb)
```

```
        out = self.conv2(out)
        out = self.res1(out) + out
        out = self.conv3(out)
        out = self.conv4(out)
        out = self.res2(out) + out
        out = self.classifier(out)
        return out
```

[16]:
```
from torchvision import models
from torchvision.models import resnet18

class DistortionClassifier18(ImageClassificationBase):
    def __init__(self, num_classes):
        super().__init__()
        self.network = models.resnet18()
        self.network.fc = nn.Linear(self.network.fc.in_features, num_classes)

    def forward(self, xb):
        return self.network(xb)
```

[17]:
```
model = to_device(DistortionClassifier18(5), device)
model
```

[17]:
```
DistortionClassifier18(
  (network): ResNet(
    (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (relu): ReLU(inplace=True)
    (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1,
ceil_mode=False)
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
      (1): BasicBlock(
        (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
        (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer2): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer3): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
```

```
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
      )
    )
    (layer4): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
```

```
      )
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
    (fc): Linear(in_features=512, out_features=5, bias=True)
  )
)
```

[18]:
```
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def get_lr(optimizer):
    for param_group in optimizer.param_groups:
        return param_group['lr']

def fit_one_cycle(epochs, max_lr, model, train_loader, val_loader,␣
 ↪weight_decay=0, grad_clip=None, opt_func=torch.optim.SGD):
    torch.cuda.empty_cache()
    history = []

    # Set up custom optimizer with weight decay
    optimizer = opt_func(model.parameters(), max_lr, weight_decay=weight_decay)
    # Set up one-cycle learning rate scheduling
    sched = torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr,␣
 ↪epochs=epochs,
                                                ␣
 ↪steps_per_epoch=len(train_loader))

    for epoch in range(epochs):
        # Training phase
        model.train()
        train_losses = []
        lrs = []
        for batch in tqdm(train_loader):
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()

            # Gradient clipping
            if grad_clip:
                nn.utils.clip_grad_value_(model.parameters(), grad_clip)

            optimizer.step()
            optimizer.zero_grad()
```

```
            # Record and update learning rate
            lrs.append(get_lr(optimizer))
            sched.step()

        # Validation phase
        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        result['lrs'] = lrs
        model.epoch_end(epoch, result)
        history.append(result)
    return history
```

[21]:
```
history = [evaluate(model, valid_dl)]
history
```

[21]:
```
[{'val_loss': 1.9271801710128784, 'val_acc': 0.18359375}]
```

[19]:
```
history = []
epochs = 30
max_lr = 0.01
grad_clip = 0.1
weight_decay = 1e-4
opt_func = torch.optim.Adam
```

[24]:
```
%%time
history += fit_one_cycle(epochs, max_lr, model, train_dl, valid_dl,
                         grad_clip=grad_clip,
                         weight_decay=weight_decay,
                         opt_func=opt_func)
```

```
100%|
  | 1230/1230 [01:52<00:00, 10.96it/s]

Epoch [0], last_lr: 0.00069, train_loss: 0.0354,val_loss: 0.5561, val_acc:
0.8723

100%|
  | 1230/1230 [01:47<00:00, 11.41it/s]

Epoch [1], last_lr: 0.00152, train_loss: 0.0336,val_loss: 0.6916, val_acc:
0.7876

100%|
  | 1230/1230 [01:47<00:00, 11.40it/s]

Epoch [2], last_lr: 0.00280, train_loss: 0.0401,val_loss: 0.6363, val_acc:
0.8089

100%|
  | 1230/1230 [01:48<00:00, 11.37it/s]
```

```
Epoch [3], last_lr: 0.00437, train_loss: 0.0377,val_loss: 0.5536, val_acc:
0.7935

100%|
  | 1230/1230 [01:48<00:00, 11.30it/s]

Epoch [4], last_lr: 0.00603, train_loss: 0.0376,val_loss: 0.4903, val_acc:
0.8391

100%|
  | 1230/1230 [01:48<00:00, 11.29it/s]

Epoch [5], last_lr: 0.00760, train_loss: 0.0350,val_loss: 0.4997, val_acc:
0.8700

100%|
  | 1230/1230 [01:48<00:00, 11.37it/s]

Epoch [6], last_lr: 0.00888, train_loss: 0.0374,val_loss: 0.7302, val_acc:
0.7943

100%|
  | 1230/1230 [01:46<00:00, 11.55it/s]

Epoch [7], last_lr: 0.00971, train_loss: 0.0383,val_loss: 0.4035, val_acc:
0.8584

100%|
  | 1230/1230 [01:48<00:00, 11.35it/s]

Epoch [8], last_lr: 0.01000, train_loss: 0.0353,val_loss: 1.0227, val_acc:
0.7544

100%|
  | 1230/1230 [01:47<00:00, 11.43it/s]

Epoch [9], last_lr: 0.00994, train_loss: 0.0308,val_loss: 0.2939, val_acc:
0.8990

100%|
  | 1230/1230 [01:47<00:00, 11.46it/s]

Epoch [10], last_lr: 0.00978, train_loss: 0.0338,val_loss: 0.8697, val_acc:
0.7615

100%|
  | 1230/1230 [02:13<00:00,  9.23it/s]

Epoch [11], last_lr: 0.00950, train_loss: 0.0315,val_loss: 0.2272, val_acc:
0.9158

100%|
  | 1230/1230 [01:48<00:00, 11.31it/s]

Epoch [12], last_lr: 0.00913, train_loss: 0.0274,val_loss: 0.3721, val_acc:
0.8663
```

```
100%|
  | 1230/1230 [01:47<00:00, 11.47it/s]

Epoch [13], last_lr: 0.00867, train_loss: 0.0286,val_loss: 0.4039, val_acc:
0.8756

100%|
  | 1230/1230 [01:47<00:00, 11.42it/s]

Epoch [14], last_lr: 0.00812, train_loss: 0.0268,val_loss: 0.3494, val_acc:
0.8897

100%|
  | 1230/1230 [01:47<00:00, 11.48it/s]

Epoch [15], last_lr: 0.00750, train_loss: 0.0254,val_loss: 0.9066, val_acc:
0.7996

100%|
  | 1230/1230 [01:49<00:00, 11.19it/s]

Epoch [16], last_lr: 0.00683, train_loss: 0.0229,val_loss: 0.3452, val_acc:
0.8859

100%|
  | 1230/1230 [01:54<00:00, 10.74it/s]

Epoch [17], last_lr: 0.00611, train_loss: 0.0210,val_loss: 0.8006, val_acc:
0.7866

100%|
  | 1230/1230 [01:52<00:00, 10.90it/s]

Epoch [18], last_lr: 0.00537, train_loss: 0.0190,val_loss: 0.6987, val_acc:
0.8074

100%|
  | 1230/1230 [01:48<00:00, 11.31it/s]

Epoch [19], last_lr: 0.00463, train_loss: 0.0183,val_loss: 0.3725, val_acc:
0.8914

100%|
  | 1230/1230 [01:49<00:00, 11.19it/s]

Epoch [20], last_lr: 0.00389, train_loss: 0.0164,val_loss: 0.5377, val_acc:
0.8873

100%|
  | 1230/1230 [01:49<00:00, 11.27it/s]

Epoch [21], last_lr: 0.00317, train_loss: 0.0146,val_loss: 0.2741, val_acc:
0.9070

100%|
  | 1230/1230 [01:48<00:00, 11.33it/s]
```

```
Epoch [22], last_lr: 0.00250, train_loss: 0.0131,val_loss: 0.2369, val_acc:
0.9158

100%|
  | 1230/1230 [01:47<00:00, 11.40it/s]

Epoch [23], last_lr: 0.00188, train_loss: 0.0108,val_loss: 0.4028, val_acc:
0.8691

100%|
  | 1230/1230 [01:47<00:00, 11.44it/s]

Epoch [24], last_lr: 0.00133, train_loss: 0.0086,val_loss: 0.3208, val_acc:
0.9123

100%|
  | 1230/1230 [01:47<00:00, 11.44it/s]

Epoch [25], last_lr: 0.00087, train_loss: 0.0079,val_loss: 0.4474, val_acc:
0.8558

100%|
  | 1230/1230 [01:47<00:00, 11.41it/s]

Epoch [26], last_lr: 0.00050, train_loss: 0.0064,val_loss: 0.4487, val_acc:
0.8563

100%|
  | 1230/1230 [01:47<00:00, 11.43it/s]

Epoch [27], last_lr: 0.00022, train_loss: 0.0056,val_loss: 0.3277, val_acc:
0.8937

100%|
  | 1230/1230 [01:47<00:00, 11.48it/s]

Epoch [28], last_lr: 0.00006, train_loss: 0.0049,val_loss: 0.4415, val_acc:
0.8615

100%|
  | 1230/1230 [01:48<00:00, 11.38it/s]

Epoch [29], last_lr: 0.00000, train_loss: 0.0044,val_loss: 0.3111, val_acc:
0.9030
CPU times: total: 14min 27s
Wall time: 56min 41s
```

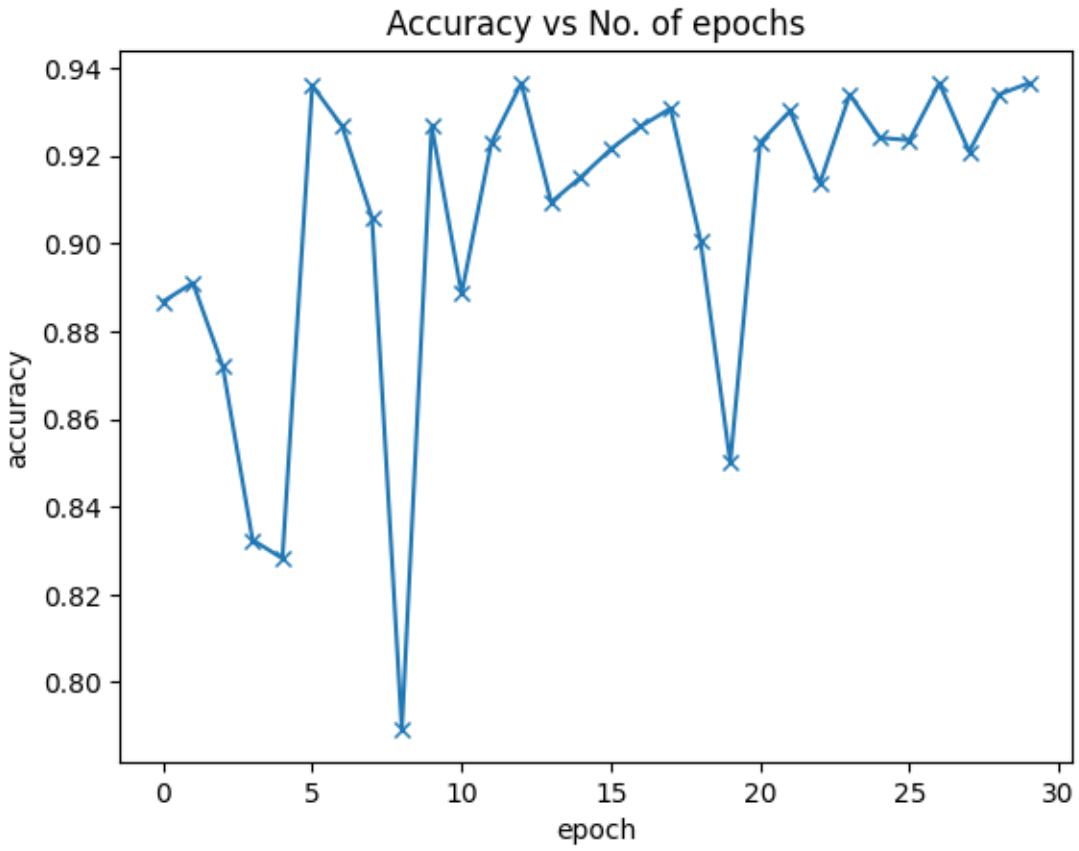
[40]:
```
torch.cuda.empty_cache()
```

0.0.5 Save Model

[50]:
```
torch.save(model.state_dict(), "distortion_classifier_resnet18-pretrained-90.
↪pt")
```

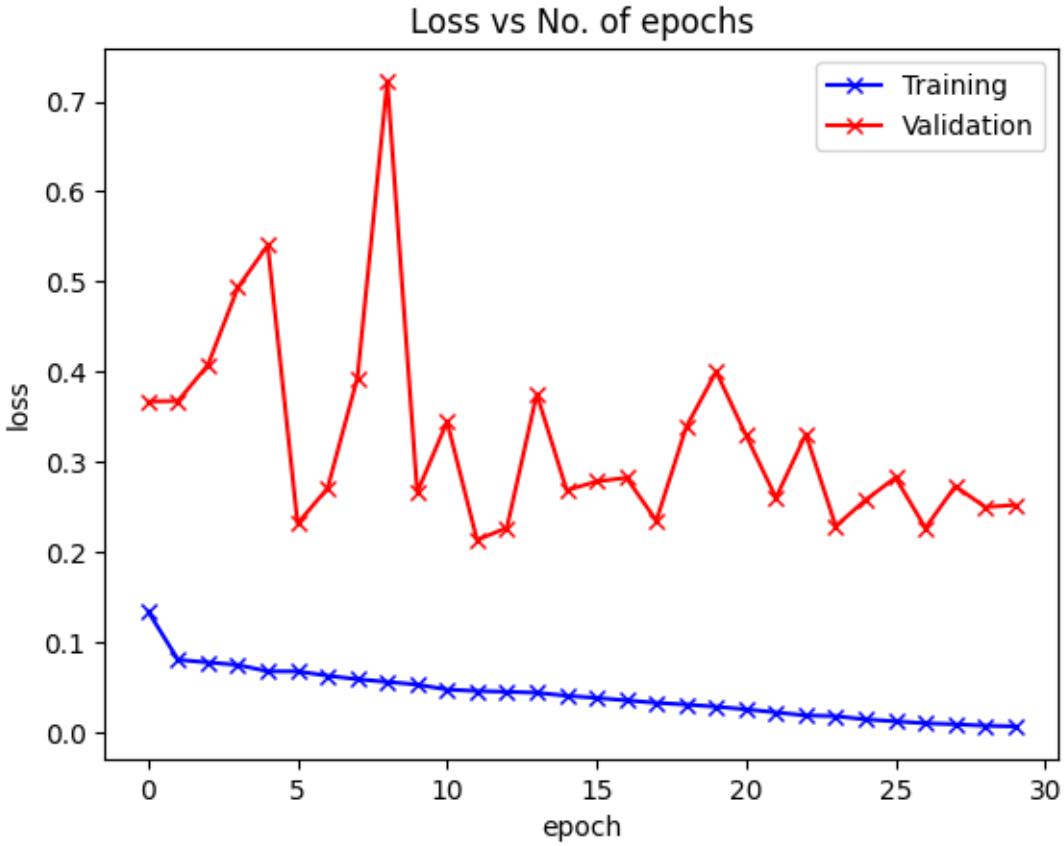
0.0.6 Performance Graph

[46]:
```
def plot_accuracies(history):
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs No. of epochs')

plot_accuracies(history)
```

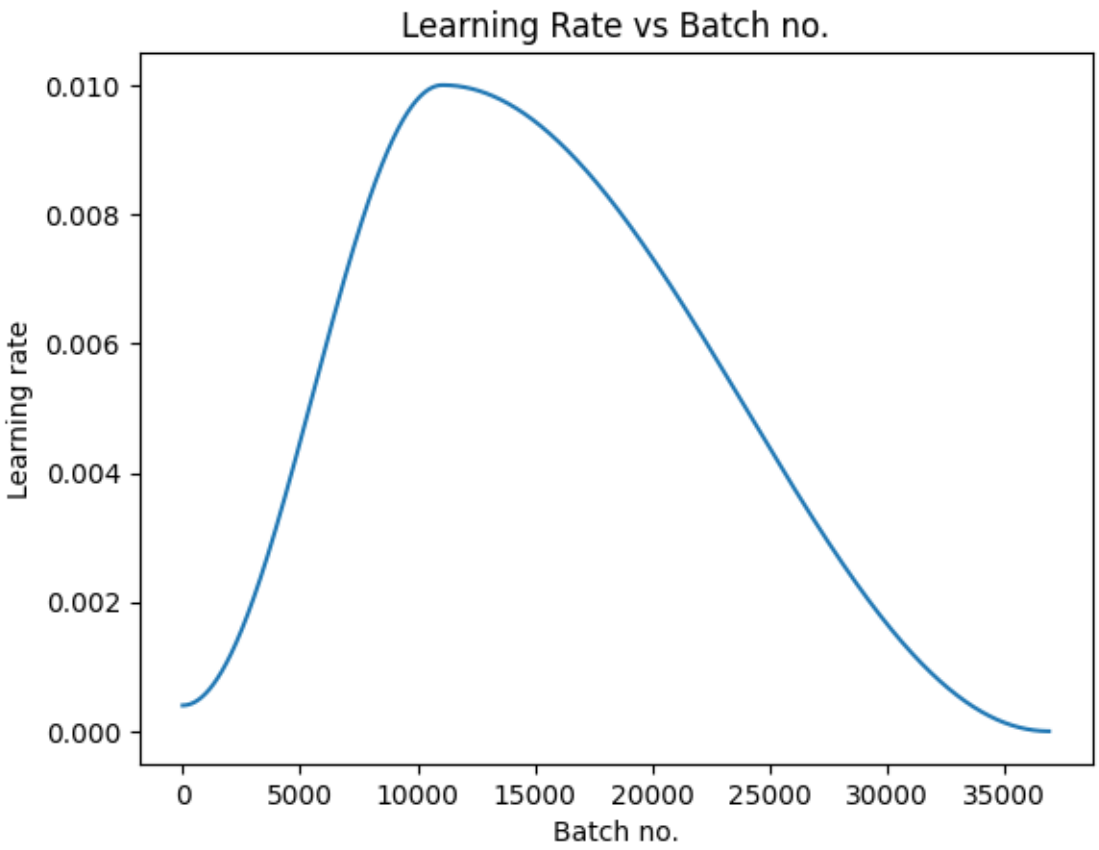


[48]:
```
def plot_losses(history):
    train_losses = [x['train_loss'] for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Validation'])
    plt.title('Loss vs No. of epochs')
```

```
plot_losses(history)
```



[50]:
```
def plot_lrs(history):
    lrs = np.concatenate([x.get('lrs', []) for x in history])
    plt.plot(lrs)
    plt.xlabel('Batch no.')
    plt.ylabel('Learning rate')
    plt.title('Learning Rate vs Batch no.')

plot_lrs(history)
```



0.0.7 Evaluate Model

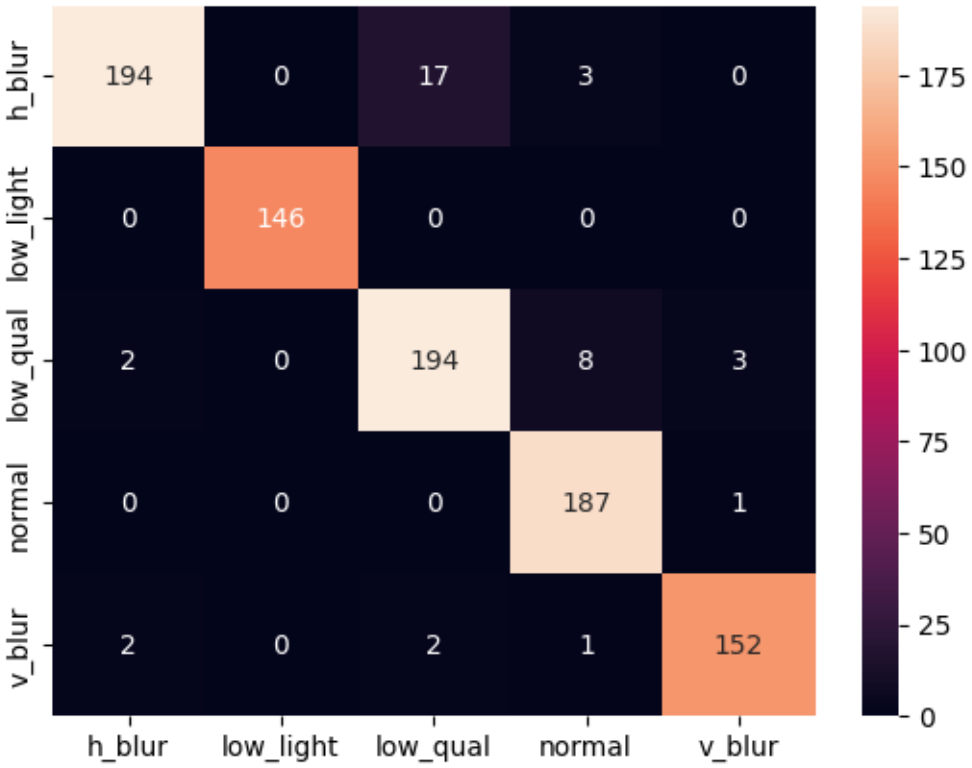
[45]:
```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
```

[46]:
```
all_preds = []
all_labels = []

model.eval()
with torch.no_grad():
    for images, labels in valid_dl:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)

        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())
```

[47]:
```
class_names = train_ds.classes

print("Classification Report:")
print(classification_report(all_labels, all_preds, target_names=class_names))

print("\nConfusion Matrix:")
sns.heatmap(confusion_matrix(all_labels, all_preds), annot=True, fmt='d',␣
 ↪xticklabels=class_names, yticklabels=class_names)
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

      h_blur       0.98      0.91      0.94       214
   low_light       1.00      1.00      1.00       146
    low_qual       0.91      0.94      0.92       207
      normal       0.94      0.99      0.97       188
      v_blur       0.97      0.97      0.97       157

    accuracy                           0.96       912
   macro avg       0.96      0.96      0.96       912
weighted avg       0.96      0.96      0.96       912
```

Confusion Matrix:

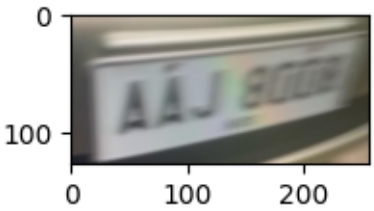


0.0.8 Predict Image

[36]:
```
def predict_image(img, model):
    # Convert to a batch of 1
    xb = to_device(img.unsqueeze(0), device)
    # Get predictions from the model
    yb = model(xb)
    # Pick index with the highest probability
    _, preds = torch.max(yb, dim=1)
    # Retrieve the class label
    return train_ds.classes[preds[0].item()]
```

[64]:
```
img, label = valid_ds[0]
denorm_img = denormalize(img, *stats)[0].permute(1, 2, 0)
plt.figure(figsize=(2,20))
plt.imshow(denorm_img)
print(denorm_img.shape)
print(f'Label: {train_ds.classes[label]}, Predicted: {predict_image(img,␣
↪model)}')
```

```
torch.Size([128, 256, 3])
Label: h_blur, Predicted: h_blur
```



[88]:
```
img, label = valid_ds[1010]
denorm_img = denormalize(img, *stats)[0].permute(1, 2, 0)
plt.figure(figsize=(2,20))
plt.imshow(denorm_img)
print(denorm_img.shape)
print(f'Label: {train_ds.classes[label]}, Predicted: {predict_image(img,␣
↪model)}')
```

```
torch.Size([128, 256, 3])
Label: normal, Predicted: normal
```

[152]:
```
img, label = test_ds[37]
denorm_img = denormalize(img, *stats)[0].permute(1, 2, 0)
plt.figure(figsize=(2,20))
plt.imshow(denorm_img)
print(denorm_img.shape)
print(f'Label: {train_ds.classes[label]}, Predicted: {predict_image(img,␣
↪model)}')
```

```
torch.Size([128, 256, 3])
Label: h_blur, Predicted: normal
```

[70]:
```
[evaluate(model, valid_dl)]
```

[70]:
```
[{'val_loss': 0.3463773727416992, 'val_acc': 0.9001893997192383}]
```

0.0.9 Test with images outside the dataset

[42]:
```
from PIL import Image
```

[134]:
```
h_blurs = os.listdir(os.path.join(DATA_DIR, 'test3', 'low_qual'))

for image_name in h_blurs:
    external_img = Image.open(os.path.join(DATA_DIR, 'test3',␣
↪'low_qual',image_name)).convert('RGB')
    img_tensor = transform(external_img)

    prediction = predict_image(img_tensor, model)
    if prediction != 'low_qual':
```

```
        print(image_name)
```

```
11.jpg
15.jpg
20.jpg
23.jpg
26.jpg
4.jpg
IMG_3689.PNG
IMG_3690.PNG
IMG_3691.PNG
IMG_3751.PNG
IMG_3752.PNG
```

```
---------------------------------------------------------------------------
UnidentifiedImageError                    Traceback (most recent call last)
Cell In[134], line 4
      1 h_blurs = os.listdir(os.path.join(DATA_DIR, 'test3', 'low_qual'))
      3 for image_name in h_blurs:
----> 4     external_img =␣
 ↪Image.open(os.path.join(DATA_DIR,          ,          ,image_name)).
 ↪convert('RGB')
      5     img_tensor = transform(external_img)
      7     prediction = predict_image(img_tensor, model)

File D:\User\Jansen\Self Study\2025 - 05 -␣
 ↪MAY\LiPAD\lipad-venv\Lib\site-packages\PIL\Image.py:3572, in open(fp, mode,␣
 ↪formats)
   3570     warnings.warn(message)
   3571 msg = "cannot identify image file %r" % (filename if filename else fp)
-> 3572 raise UnidentifiedImageError(msg)

UnidentifiedImageError: cannot identify image file 'C:
 ↪\\Thesis\\LiPAD\\data\\cnn\\test3\\low_qual\\IMG_3779(1).HEIC'
```

[44]:
```
transform = T.Compose([
    T.Resize([128, 256]),
    T.ToTensor(),
    T.Normalize(*stats)
])

model.eval()
img_path = os.path.join(DATA_DIR, 'test', 'h_blur', '67.jpg')
external_img = Image.open(img_path).convert('RGB')
img_tensor = transform(external_img)
```

[112]:
```
prediction = predict_image(img_tensor, model)
print(f'Predicted distortion: {prediction}')

denorm_img = denormalize(img_tensor, *stats)[0].permute(1, 2, 0)
plt.imshow(denorm_img)
plt.axis('off')
plt.title(f'Prediction: {prediction}')
plt.show()
```

```
Predicted distortion: low_qual
```

0.0.10 Load the model using .pt

[38]:
```
model = DistortionClassifier18(5)
model_wts = torch.load('distortion_classifier.pt')
model.load_state_dict(model_wts)
model = to_device(model, device)
```

[45]:
```
[evaluate(model, valid_dl)]
```

[45]:
```
[{'val_loss': 0.010042956098914146, 'val_acc': 0.9973958134651184}]
```

[81]:
```
torch.cuda.empty_cache()
```

0.0.11 Test with test dataset

[55]:
```
test_ds = ImageFolder(os.path.join(DATA_DIR, 'test'), valid_tfms)
test_dl = DataLoader(test_ds, batch_size*5, num_workers=3, pin_memory=True)
test_dl = DeviceDataLoader(test_dl, device)
```

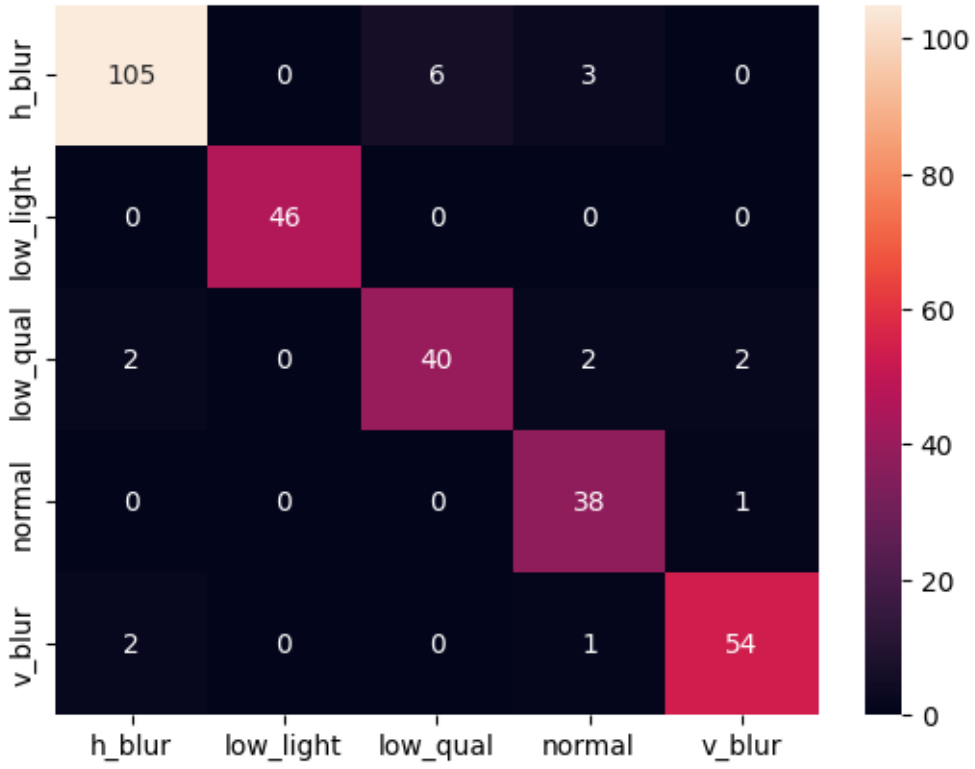
[56]:
```
[evaluate(model, test_dl)]
```

[56]:
```
[{'val_loss': 0.25383108854293823, 'val_acc': 0.9370861053466797}]
```

[172]:
```
all_preds = []
all_labels = []

model.eval()
with torch.no_grad():
    for images, labels in test_dl:
        images = images.to(device)
        labels = labels.to(device)
        outputs = model(images)
        _, preds = torch.max(outputs, 1)

        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

class_names = train_ds.classes

print("Classification Report:")
print(classification_report(all_labels, all_preds, target_names=class_names))

print("\nConfusion Matrix:")
sns.heatmap(confusion_matrix(all_labels, all_preds), annot=True, fmt='d',␣
 ↪xticklabels=class_names, yticklabels=class_names)
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

      h_blur       0.96      0.92      0.94       114
   low_light       1.00      1.00      1.00        46
    low_qual       0.87      0.87      0.87        46
      normal       0.86      0.97      0.92        39
      v_blur       0.95      0.95      0.95        57

    accuracy                           0.94       302
   macro avg       0.93      0.94      0.93       302
weighted avg       0.94      0.94      0.94       302
```

Confusion Matrix:



1 Transfer Learning

[31]:
```
from torchvision import models
from torchvision.models import resnet18, ResNet18_Weights

class DistortionClassifier18(ImageClassificationBase):
    def __init__(self, num_classes):
        super().__init__()
        #weights = ResNet18_Weights.DEFAULT
        self.network = models.resnet18()
        self.network.fc = nn.Linear(self.network.fc.in_features, num_classes)

    def forward(self, xb):
        return self.network(xb)
```

[33]:
```
model2 = DistortionClassifier18(10)
to_device(model2, device);
```

[247]:
```
epochs = 20
max_lr = 0.01
grad_clip = 0.1
weight_decay = 1e-4
opt_func = torch.optim.Adam
```

[251]:
```
%%time
history2 = fit_one_cycle(epochs, max_lr, model2, train_dl, valid_dl,
                         grad_clip=grad_clip,
                         weight_decay=weight_decay,
                         opt_func=opt_func)
```

```
100%|
   | 929/929 [01:42<00:00,  9.09it/s]

Epoch [0], last_lr: 0.00104, train_loss: 0.1258,val_loss: 0.2412, val_acc:
0.9380

100%|
   | 929/929 [01:40<00:00,  9.25it/s]

Epoch [1], last_lr: 0.00280, train_loss: 0.1270,val_loss: 0.2264, val_acc:
0.9372

100%|
   | 929/929 [01:40<00:00,  9.28it/s]

Epoch [2], last_lr: 0.00520, train_loss: 0.1468,val_loss: 0.3524, val_acc:
0.9017

100%|
   | 929/929 [01:40<00:00,  9.22it/s]

Epoch [3], last_lr: 0.00760, train_loss: 0.1496,val_loss: 0.2503, val_acc:
0.9252

100%|
   | 929/929 [01:40<00:00,  9.24it/s]

Epoch [4], last_lr: 0.00936, train_loss: 0.1424,val_loss: 0.7712, val_acc:
0.8380

100%|
   | 929/929 [01:45<00:00,  8.79it/s]

Epoch [5], last_lr: 0.01000, train_loss: 0.1470,val_loss: 0.2762, val_acc:
0.9431

100%|
   | 929/929 [01:45<00:00,  8.82it/s]

Epoch [6], last_lr: 0.00987, train_loss: 0.1359,val_loss: 0.2224, val_acc:
0.9363
```

```
100%|
   | 929/929 [01:44<00:00,  8.91it/s]

Epoch [7], last_lr: 0.00950, train_loss: 0.1272,val_loss: 0.2191, val_acc:
0.9357

100%|
   | 929/929 [01:43<00:00,  8.95it/s]

Epoch [8], last_lr: 0.00891, train_loss: 0.1187,val_loss: 0.1375, val_acc:
0.9603

100%|
   | 929/929 [01:43<00:00,  8.99it/s]

Epoch [9], last_lr: 0.00812, train_loss: 0.1105,val_loss: 0.1477, val_acc:
0.9572

100%|
   | 929/929 [01:45<00:00,  8.80it/s]

Epoch [10], last_lr: 0.00717, train_loss: 0.1007,val_loss: 0.1843, val_acc:
0.9452

100%|
   | 929/929 [01:45<00:00,  8.81it/s]

Epoch [11], last_lr: 0.00611, train_loss: 0.0907,val_loss: 0.1898, val_acc:
0.9521

100%|
   | 929/929 [01:45<00:00,  8.80it/s]

Epoch [12], last_lr: 0.00500, train_loss: 0.0820,val_loss: 0.2127, val_acc:
0.9503

100%|
   | 929/929 [01:45<00:00,  8.77it/s]

Epoch [13], last_lr: 0.00389, train_loss: 0.0754,val_loss: 0.1438, val_acc:
0.9594

100%|
   | 929/929 [01:46<00:00,  8.69it/s]

Epoch [14], last_lr: 0.00283, train_loss: 0.0641,val_loss: 0.1024, val_acc:
0.9740

100%|
   | 929/929 [01:47<00:00,  8.67it/s]

Epoch [15], last_lr: 0.00188, train_loss: 0.0535,val_loss: 0.0978, val_acc:
0.9736

100%|
   | 929/929 [01:47<00:00,  8.66it/s]
```

```
Epoch [16], last_lr: 0.00109, train_loss: 0.0449,val_loss: 0.0994, val_acc:
0.9751

100%|
   | 929/929 [01:45<00:00,  8.77it/s]

Epoch [17], last_lr: 0.00050, train_loss: 0.0380,val_loss: 0.0862, val_acc:
0.9796

100%|
   | 929/929 [01:45<00:00,  8.81it/s]

Epoch [18], last_lr: 0.00013, train_loss: 0.0326,val_loss: 0.0806, val_acc:
0.9797

100%|
   | 929/929 [01:46<00:00,  8.76it/s]

Epoch [19], last_lr: 0.00000, train_loss: 0.0307,val_loss: 0.0807, val_acc:
0.9799
CPU times: total: 8min 43s
Wall time: 39min 56s
```

[295]:
```
torch.save(model2.state_dict(), "distortion_classifier_resnet18.pt")
```

[1]:
```
!jupyter nbconvert --to pdf distortion_classifier.ipynb
```

```
[NbConvertApp] Converting notebook distortion_classifier.ipynb to pdf
[NbConvertApp] ERROR | Error while converting 'distortion_classifier.ipynb'
Traceback (most recent call last):
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\nbconvertapp.py", line 487, in export_single_notebook
    output, resources = self.exporter.from_filename(
                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\templateexporter.py", line 390, in from_filename
    return super().from_filename(filename, resources, **kw)  #
type:ignore[return-value]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\exporter.py", line 201, in from_filename
    return self.from_file(f, resources=resources, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\templateexporter.py", line 396, in from_file
    return super().from_file(file_stream, resources, **kw)  #
type:ignore[return-value]
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\exporter.py", line 220, in from_file
    return self.from_notebook_node(
```

```
          ^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\pdf.py", line 184, in from_notebook_node
    latex, resources = super().from_notebook_node(nb, resources=resources, **kw)
                       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\latex.py", line 92, in from_notebook_node
    return super().from_notebook_node(nb, resources, **kw)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\exporters\templateexporter.py", line 429, in
from_notebook_node
    output = self.template.render(nb=nb_copy, resources=resources)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\jinja2\environment.py", line 1304, in render
    self.environment.handle_exception()
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\jinja2\environment.py", line 939, in handle_exception
    raise rewrite_traceback_stack(source=source)
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\index.tex.j2", line 8, in top-level
template code
    ((* extends cell_style *))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\style_jupyter.tex.j2", line 176, in
top-level template code
    \prompt{(((prompt)))}{(((prompt_color)))}{(((execution_count)))}{(((extra_sp
ace)))}
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\base.tex.j2", line 7, in top-level
template code
    ((*- extends 'document_contents.tex.j2' -*))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\document_contents.tex.j2", line 51,
in top-level template code
    ((*- block figure scoped -*))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\display_priority.j2", line 5, in
top-level template code
    ((*- extends 'null.j2' -*))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\null.j2", line 30, in top-level
```

```
template code
    ((*- block body -*))
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\base.tex.j2", line 241, in block
'body'
    ((( super() )))
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\null.j2", line 32, in block 'body'
    ((*- block any_cell scoped -*))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\null.j2", line 85, in block
'any_cell'
    ((*- block markdowncell scoped-*)) ((*- endblock markdowncell -*))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-
venv\share\jupyter\nbconvert\templates\latex\document_contents.tex.j2", line 68,
in block 'markdowncell'
    ((( cell.source | citation2latex | strip_files_prefix |
convert_pandoc('markdown+tex_math_double_backslash', 'json',extra_args=[]) |
resolve_references | convert_explicitly_relative_paths |
convert_pandoc('json','latex'))))
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\filters\pandoc.py", line 36, in convert_pandoc
    return pandoc(source, from_format, to_format, extra_args=extra_args)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\utils\pandoc.py", line 50, in pandoc
    check_pandoc_version()
  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\utils\pandoc.py", line 98, in check_pandoc_version
    v = get_pandoc_version()
        ^^^^^^^^^^^^^^^^^^^^

  File "D:\User\Jansen\Self Study\2025 - 05 - MAY\LiPAD\lipad-venv\Lib\site-
packages\nbconvert\utils\pandoc.py", line 75, in get_pandoc_version
    raise PandocMissing()
nbconvert.utils.pandoc.PandocMissing: Pandoc wasn't found.
Please check that pandoc is installed:
https://pandoc.org/installing.html
```

[ ]: