Face-Mask-App: AI

November 25, 2020

Outline

Introduction

- Neural networks
- Two different models
  - Face model
  - Mask model
- Transfer learning
  - ResNet18
- PyTorch

Model & Training

Model

- Pretty standard transfer learning model
  - Log Softmax output
  - Negative Log Likelihood loss function
  - Stochastic Gradient Descent with Momentum
  - Cyclical learning rate
- Pretty standard stuff!

```
def get_model(dataloaders, n_epochs=30):
    device = 'cuda' if torch.cuda.is_available() else 'cpu'

    # load pretrained resnet model
    model = models.resnet18(pretrained=True)

    # replace output layer, we have two outputs
    model.fc = nn.Sequential(
        nn.Linear(model.fc.in_features, 2),
        nn.LogSoftmax(dim=1)
    )

    # transform to GPU if needed
    model = model.to(device)

    # loss function
    criterion = nn.NLLLoss()

    # Learning rate and momemtum will be overriden by scheduler
    optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

    # One Cycle Policy scheduler
    scheduler = torch.optim.lr_scheduler.OneCycleLR(
        optimizer
```

Model & Training

Training

- Standard PyTorch training loop
  - Resize images
  - Random data augmentation
- Nothing special here!

```
# Iterate over data.
for inputs, labels in dataloaders[phase]:
    inputs = inputs.to(device)
    labels = labels.to(device)

    # zero the parameter gradients
    optimizer.zero_grad()

    # forward
    # track history if only in train
    with torch.set_grad_enabled(phase == 'train'):
        outputs = model(inputs)
        loss = criterion(outputs, labels)

        # backward + optimize only if in training phase
        if phase == 'train':
            loss.backward()
            optimizer.step()
            scheduler.step()

    # statistics
    _, preds = torch.max(outputs, 1)
    running_loss += loss.item() * inputs.size(0)
    n_correct = torch.sum(preds == labels)
    running_corrects += n_correct
```

Datasets

Kaggle

- https://www.kaggle.com/alexandralorenzo/maskdetection
  - For YOLO
  - Bounding boxes
  - 588 training images
  - 167 validation images
  - 84 testing images
  - Crowd images

Datasets

Mask model

- https://www.kaggle.com/alexandralorenzo/maskdetection
  - For YOLO
  - Bounding boxes
  - 588 training images
  - 167 validation images
  - 84 testing images
  - Crowd images

Messy - use as a base!

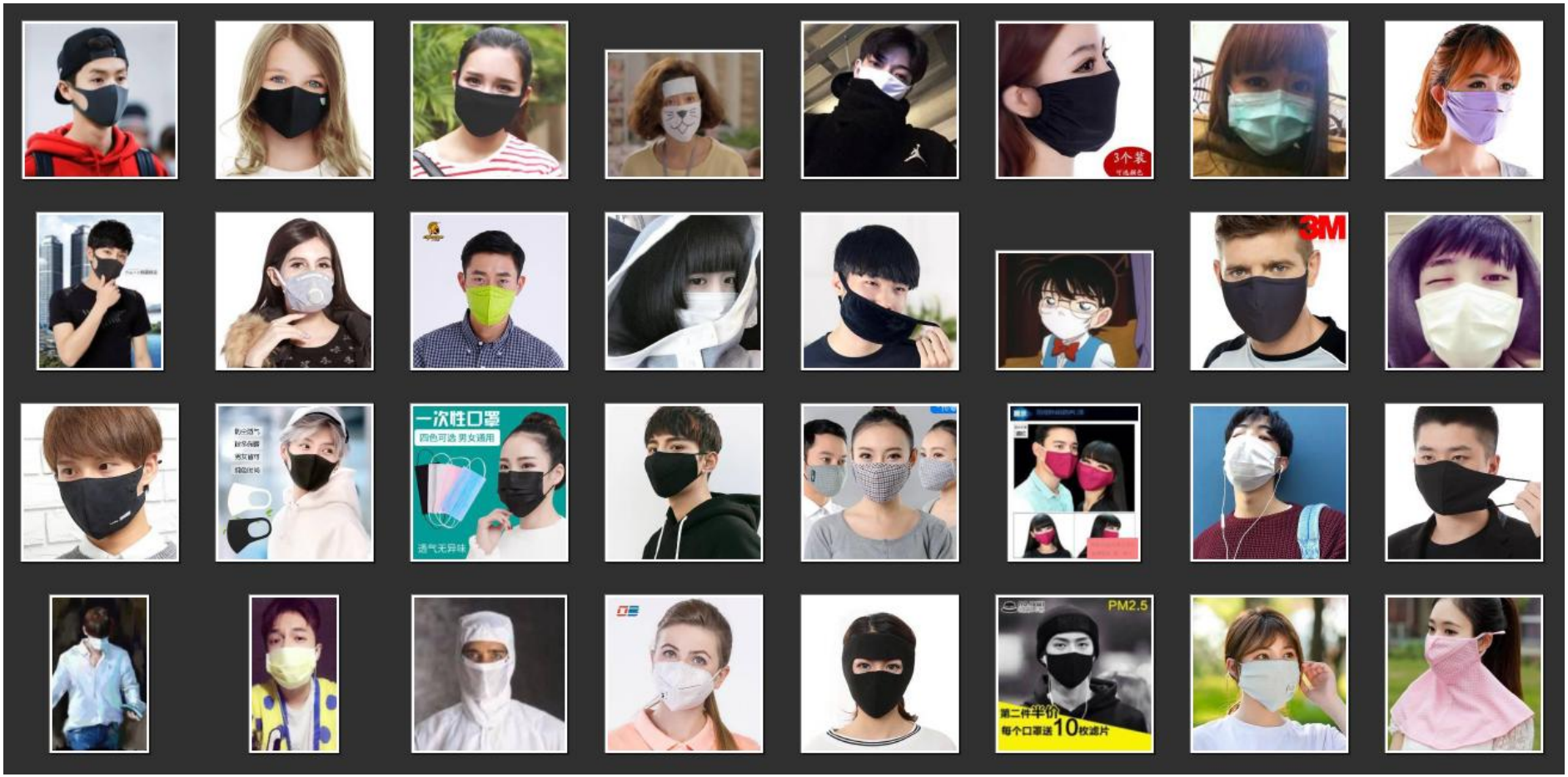
Datasets

Mask model

- Organize to PyTorch image folder structure
- Collect more data
- Training set
  - 408 images with mask
  - 432 images without mask
- Validation set
  - 80 images with mask
  - 94 images without mask
- Testing set
  - Use own data only!
  - 30 images with mask
  - 29 images without mask
  - Hundreds of tests in practice

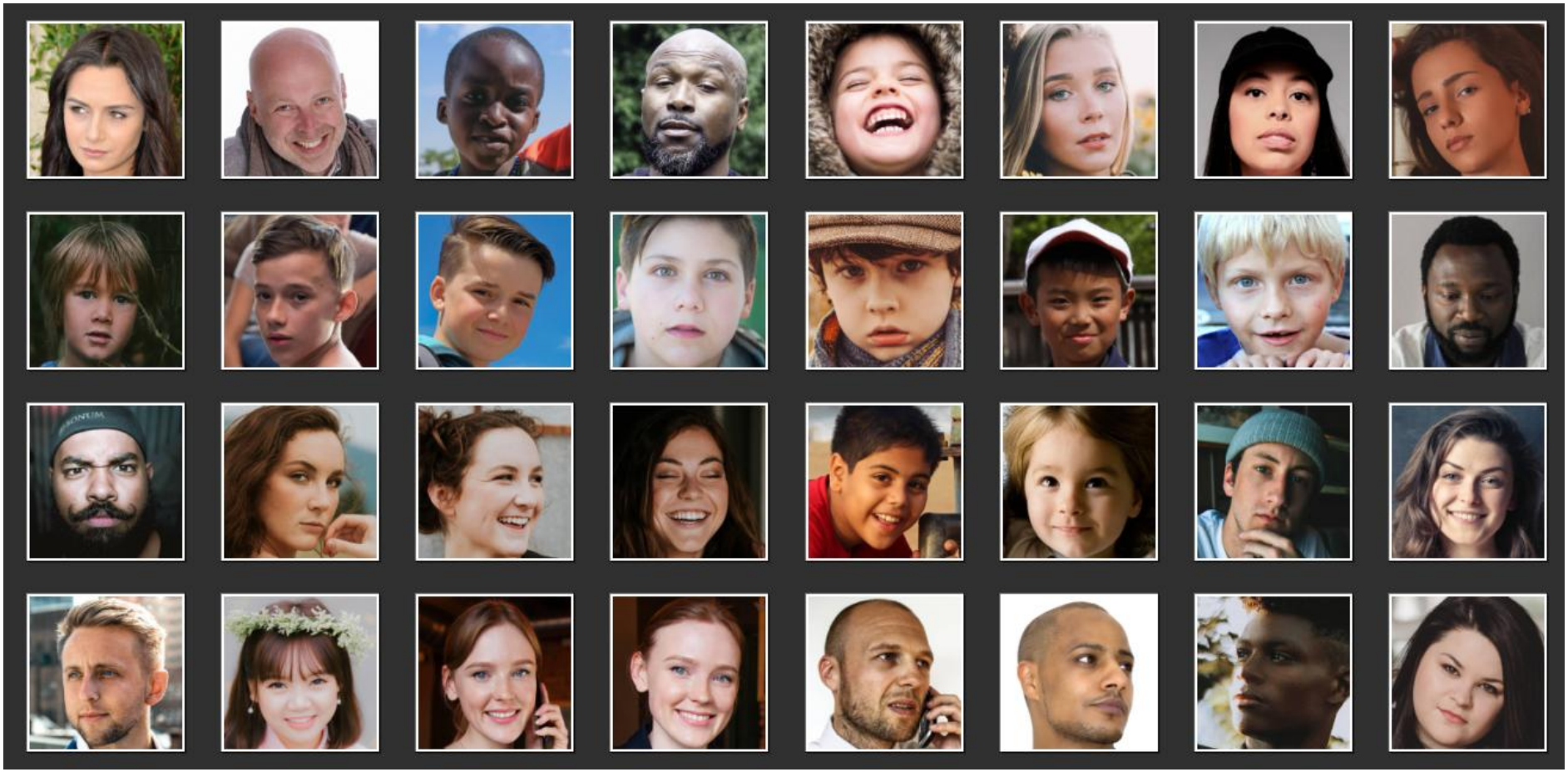
Datasets

Mask model



Datasets

Mask model



Datasets

Face model

- Collect all images with and without masks to another catogery
- Collect images without faces to another category
- Also custom images without faces
  - Take pictures of floors, inside pockets, walls, ...

Training

Results

- Same training code
- Test on real-world data
- Mask model
  - Around 94% accuracy on test set
  - Very little false positives
- Face model
  - Around 90% accuracy on test set
  - Almost no false positives at all

Model to mobile

Code

- Only a couple lines of code
  - Load models

```
try {
    face_model = Module.load(assetFilePath(context, "face_model_mobile.pt"));
    mask_model = Module.load(assetFilePath(context, "mask_model_mobile.pt"));
```

  - Predict

```
final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(image,
        TensorImageUtils.TORCHVISION_NORM_MEAN_RGB, TensorImageUtils.TORCHVISION_NORM_STD_RGB);

final Tensor outputTensor = mask_model.forward(IValue.from(inputTensor)).toTensor();
final float[] scores = outputTensor.getDataAsFloatArray();
```

Model to mobile

Code

- Only a couple lines of code
  - Load models

```
try {
    face_model = Module.load(assetFilePath(context, "face_model_mobile.pt"));
    mask_model = Module.load(assetFilePath(context, "mask_model_mobile.pt"));
```

  - Predict

```
final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(image,
        TensorImageUtils.TORCHVISION_NORM_MEAN_RGB, TensorImageUtils.TORCHVISION_NORM_STD_RGB);

final Tensor outputTensor = mask_model.forward(IValue.from(inputTensor)).toTensor();
final float[] scores = outputTensor.getDataAsFloatArray();
```

- Niclas, our Android expert, will tell you more next!