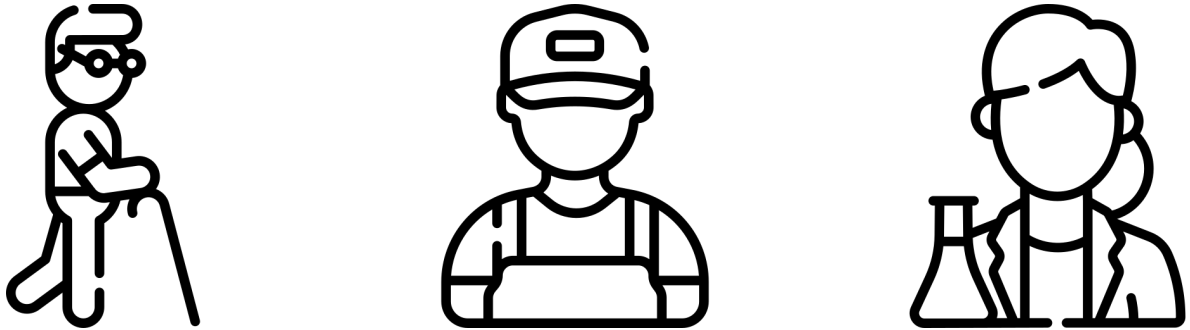
# Custom Object Classification with Pytorch Mobile

Alpay Sabuncuoğlu

# Motivation



Developing custom models increases the accessibility and productivity in many areas. Delivering AI to all is possible by employing these models using smartphones.

# Applying Transfer Learning on MobileNetv3

- MobileNet-v3 is designed for high performance using limited space and power.
- TorchVision provides a pretrained model for MobileNet-v3.

```
large = torchvision.models.mobilenet_v3_large(pretrained=True, width_mult=1.0,  reduced_tail=False, dilated=False)
small = torchvision.models.mobilenet_v3_small(pretrained=True)
quantized = torchvision.models.quantization.mobilenet_v3_large(pretrained=True)
```

# Training the last layers of MobileNet

- Change the last layers with a newly trained classifier layer.
- Freezing the previous layers, re-training the classifier layers allow us to use this model as feature extractor.
- Using only a few examples we can achieve ~0.95 accuracy depending on the dataset quality.

```
  (classifier): Sequential(
    (0): Linear(in_features=960, out_features=1280, bias=True)
    (1): Hardswish()
    (2): Dropout(p=0.2, inplace=True)
    (3): Linear(in_features=1280, out_features=1000, bias=True)
  )
```

```
for param in model_conv.parameters():
    param.requires_grad = False
# Parameters of newly constructed modules have requires_grad=True by
default
num_ftrs = model_conv.classifier[0].in_features
print(num_ftrs)
# Here the size of each output sample is set to 2.
# Alternatively, it can be generalized to nn.Linear(num_ftrs,
len(class_names)).
model_conv.classifier = nn.Linear(num_ftrs, 2)
model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()
# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(model_conv.classifier.parameters(), lr=0.001,
momentum=0.9)
# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7,
gamma=0.1)
```

# Adding Your Data Interactive on Notebook

Using the notebook, add new samples to your custom dataset. This kind of setup especially useful for personal customized models such as having a custom accessibility issue.

```
from IPython.display import Image

# complete this step for Training and Validation categories
for i in range(10):
    try:
        photo_url = "/content/drive/MyDrive/datasets/val/down/down{}.jpg".format(i);
        filename = take_photo(filename=photo_url)
        print('Saved to {}'.format(filename))
        i+=1
        # Show the image which was just taken.
        #display(Image(filename))
    except Exception as err:
        # Errors will be thrown if the user does not have a webcam or if they do not
        # grant the page permission to access it.
        print(str(err))
```

Capture

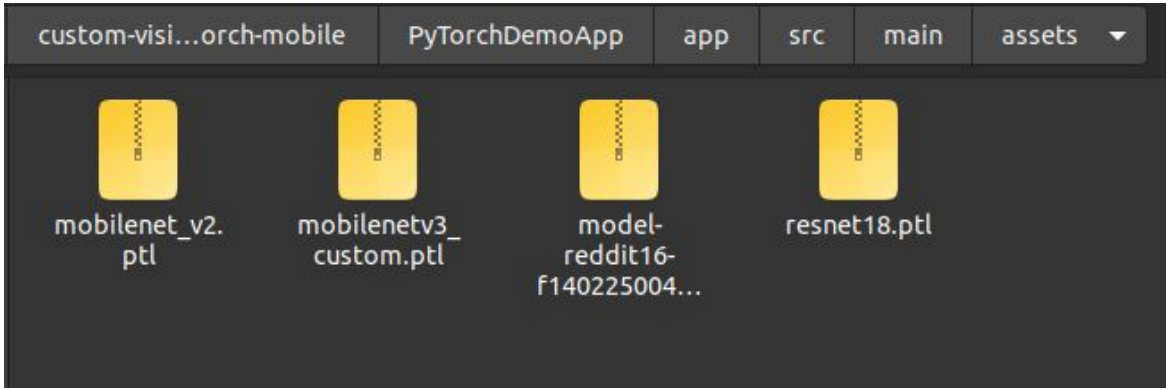
# Optimize the Model for Mobile

Notice that .ptl is compatible with torch mobile 1.9's Lite Interpreter.

```
scripted_module = torch.jit.script(model_conv)
# Export full jit version model (not compatible mobile interpreter), leave it here for comparison
scripted_module.save("mobilenetv3_custom.pt")
# Export mobile interpreter version model (compatible with mobile interpreter)
optimized_scripted_module = optimize_for_mobile(scripted_module)
optimized_scripted_module._save_for_lite_interpreter("mobilenetv3_custom.ptl")
```

# Copy the model to Assets folder

In the assets folder, you will also see the updated .ptl versions of pre-trained models with ImageNet.

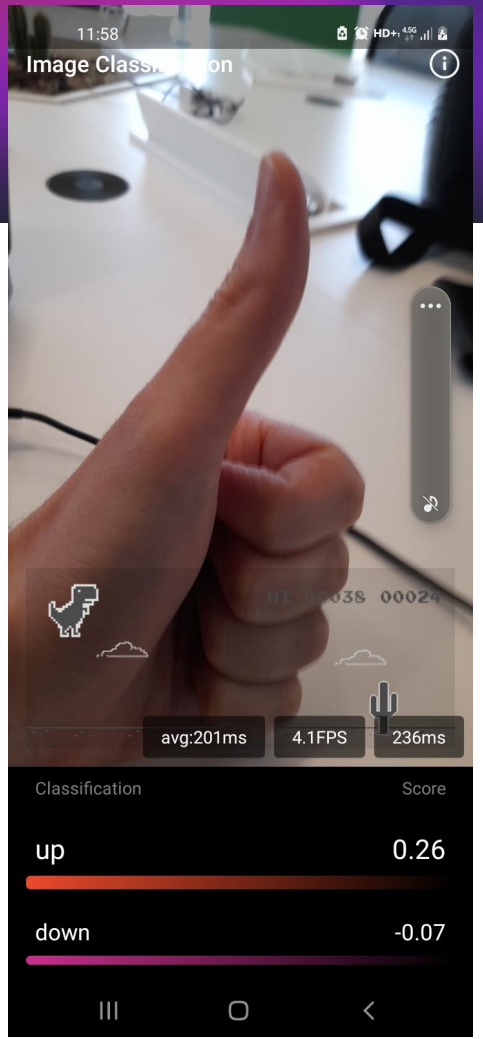


# Update the Android Demo

Now, you can create games, applications and services using these custom models.

We controlled a simple Dino game to show an example use. The simplest update is simulating key-press to control web apps:

```
if (result.topNClassNames[0].contentEquals(Constants.CUSTOM_CLASSES[0])){
   inputConnection.sendKeyEvent(
      new KeyEvent(KeyEvent.ACTION_DOWN, KeyEvent.KEYCODE_SPACE));
}
```



# What's Next? Inspire Yourself

Experiments with Google, PyTorch Mobile Demos, AllenAI Demos...