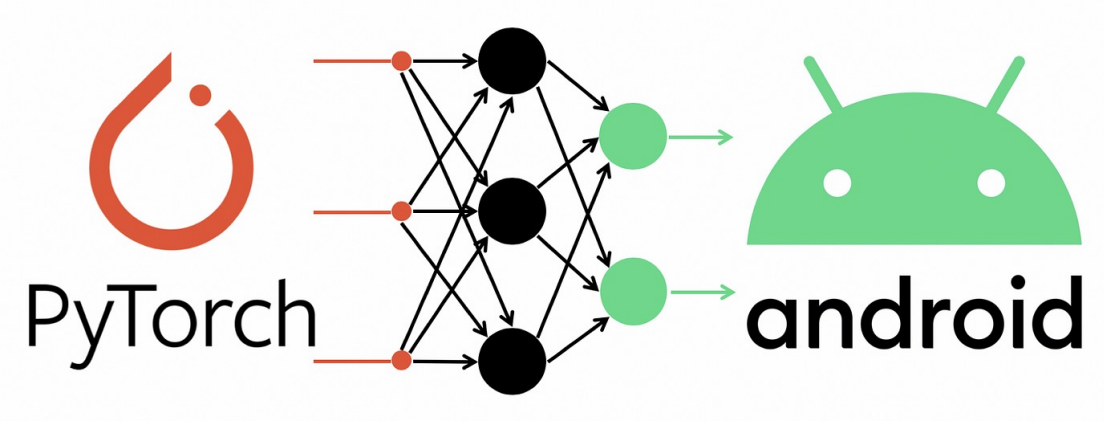
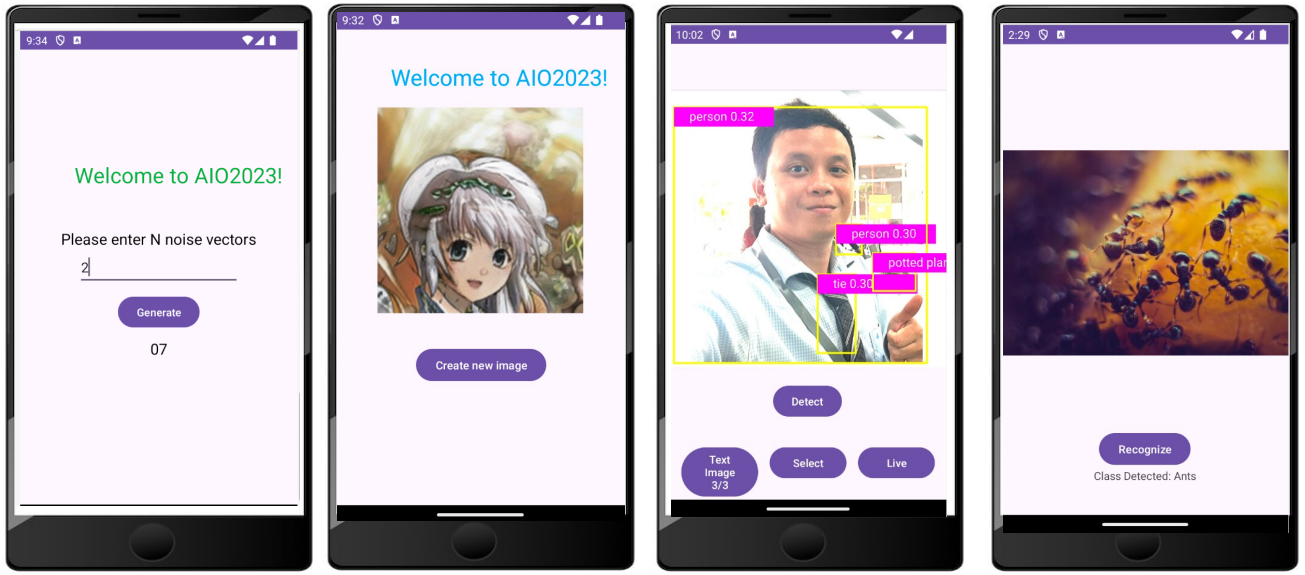
AI VIETNAM
All-in-One Course

Mobile Application Development Using AI with Pytorch By Examples





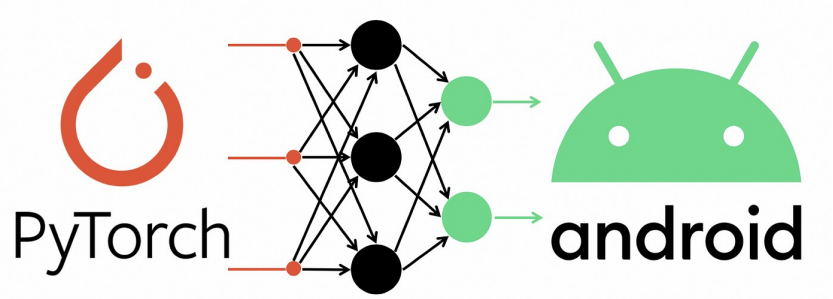
Number generator | Picture Generator | Object Detection | Object Classification

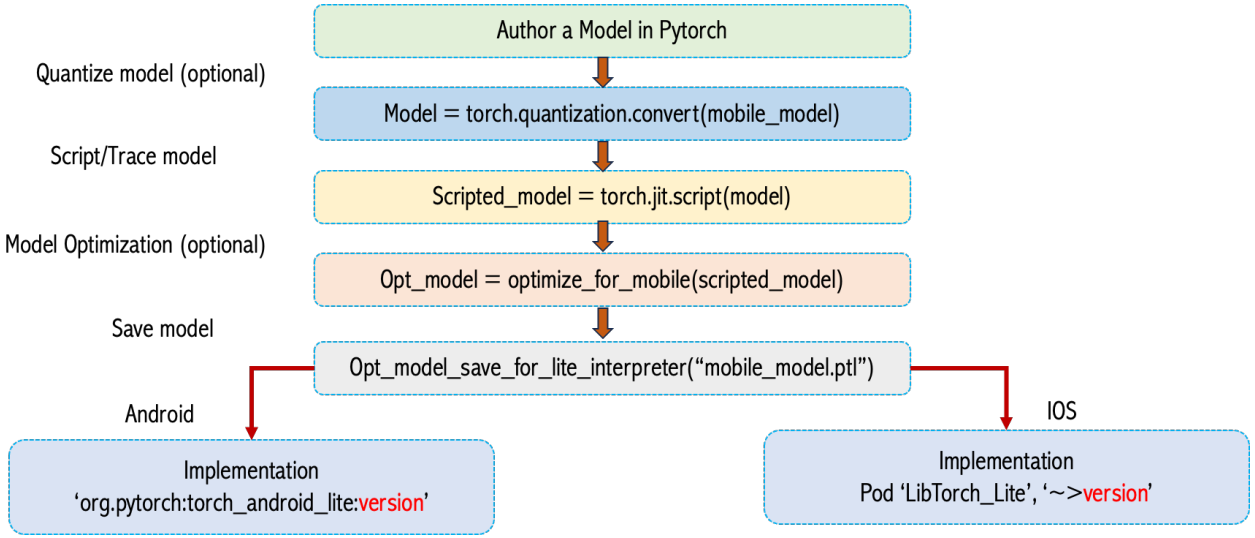
Vinh Dinh Nguyen- PhD in Computer Science

Outline

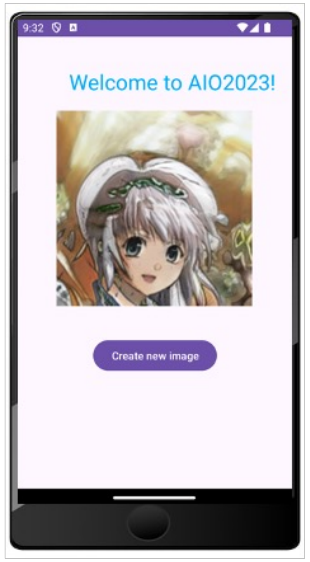
- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

Today's Contents

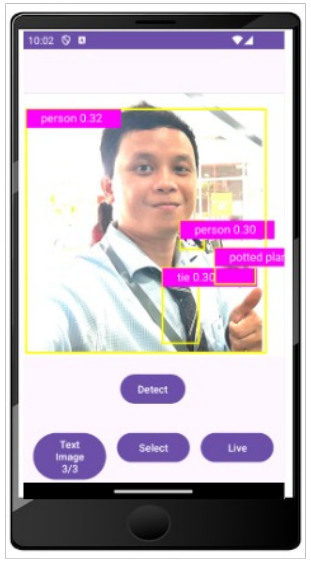




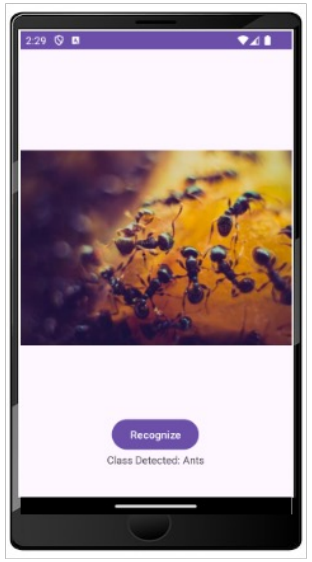
Number generator



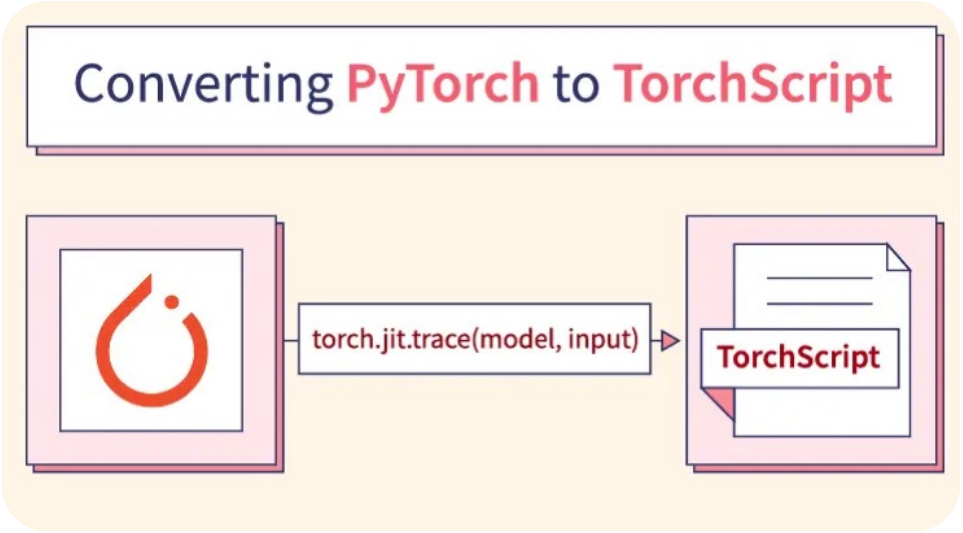
Picture Generator



Object Detection



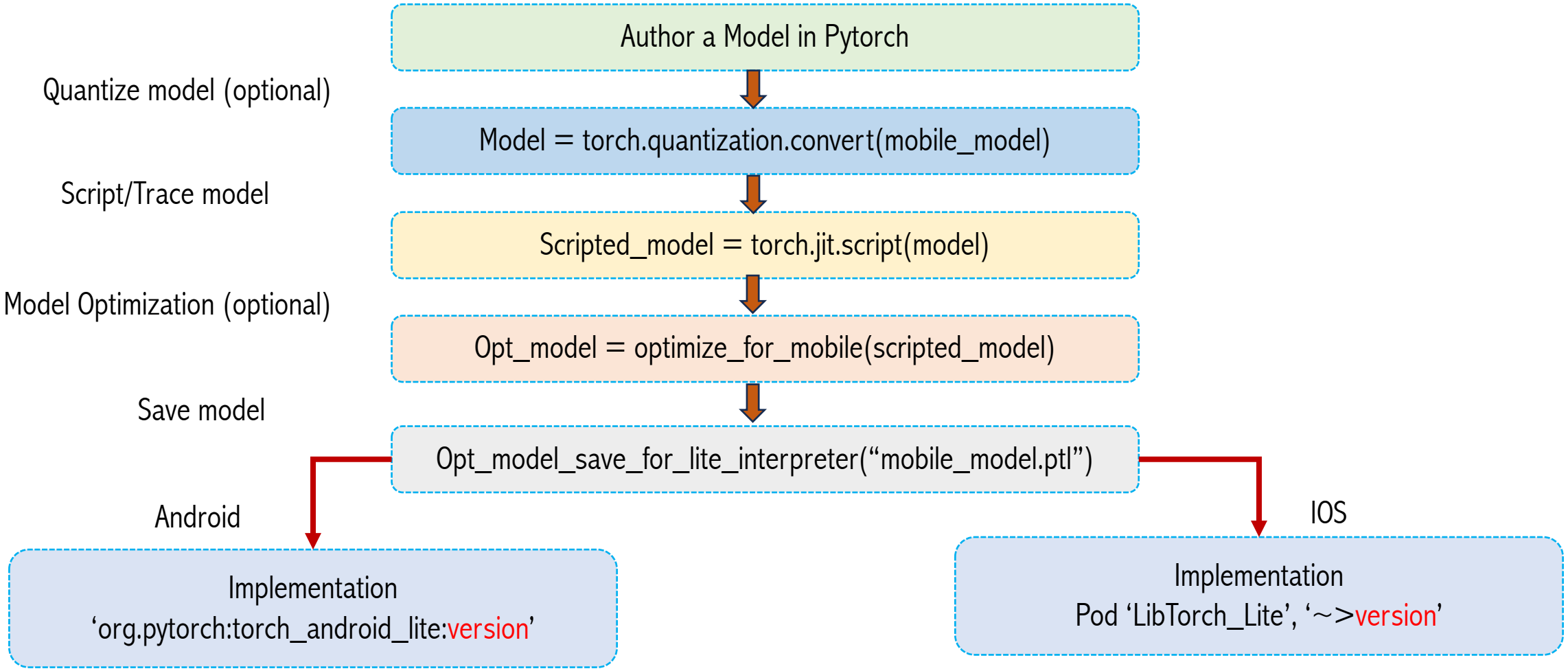
Object Classification



Outline

- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

DEPLOYMENT WORKFLOW



PyTorch JIT and TorchScript

With TorchScript, PyTorch aims to create a unified framework from research to production. TorchScript will take your PyTorch modules as input and convert them into a production-friendly format

PyTorch supports 2 separate modes to handle research and production environment.

- First is the *Eager mode.* It is built for faster prototyping, training, and experimentation.
- Second is the *Script mode.* It is focused on the production use case. It has 2 components *PyTorch JIT* and *TorchScript.*

```
# Example 1.1 Pytorch cpu version

model_ft = torchvision.models.resnet18(pretrained=True)
model_ft.eval()
x_ft = torch.rand(1,3, 224,224)
print("Pytorch cpu version", np.mean([timer(model_ft,x_ft) for _ in range(10)]))
```

PyTorch JIT and TorchScript

With TorchScript, PyTorch aims to create a unified framework from research to production. TorchScript will take your PyTorch modules as input and convert them into a production-friendly format

PyTorch supports 2 separate modes to handle research and production environment.

- First is the *Eager mode.* It is built for faster prototyping, training, and experimentation.
- Second is the *Script mode.* It is focused on the production use case. It has 2 components *PyTorch JIT* and *TorchScript.*

```
# Example 1.2 Pytorch gpu version

model_ft_gpu = torchvision.models.resnet18(pretrained=True).cuda()
x_ft_gpu = x_ft.cuda()
model_ft_gpu.eval()
print("Pytorch gpu version", np.mean([timer(model_ft_gpu,x_ft_gpu) for _ in range(10)]))
```

PyTorch JIT and TorchScript

With TorchScript, PyTorch aims to create a unified framework from research to production. TorchScript will take your PyTorch modules as input and convert them into a production-friendly format

PyTorch supports 2 separate modes to handle research and production environment.

- First is the *Eager mode.* It is built for faster prototyping, training, and experimentation.
- Second is the *Script mode.* It is focused on the production use case. It has 2 components *PyTorch JIT* and *TorchScript.*

```
# Example 2.1 torch.jit.script cpu version

script_cell = torch.jit.script(model_ft, (x_ft))
print("torch.jit.script cpu version", np.mean([timer(script_cell,x_ft) for _ in range(10)]))
```

PyTorch JIT and TorchScript

With TorchScript, PyTorch aims to create a unified framework from research to production. TorchScript will take your PyTorch modules as input and convert them into a production-friendly format

PyTorch supports 2 separate modes to handle research and production environment.

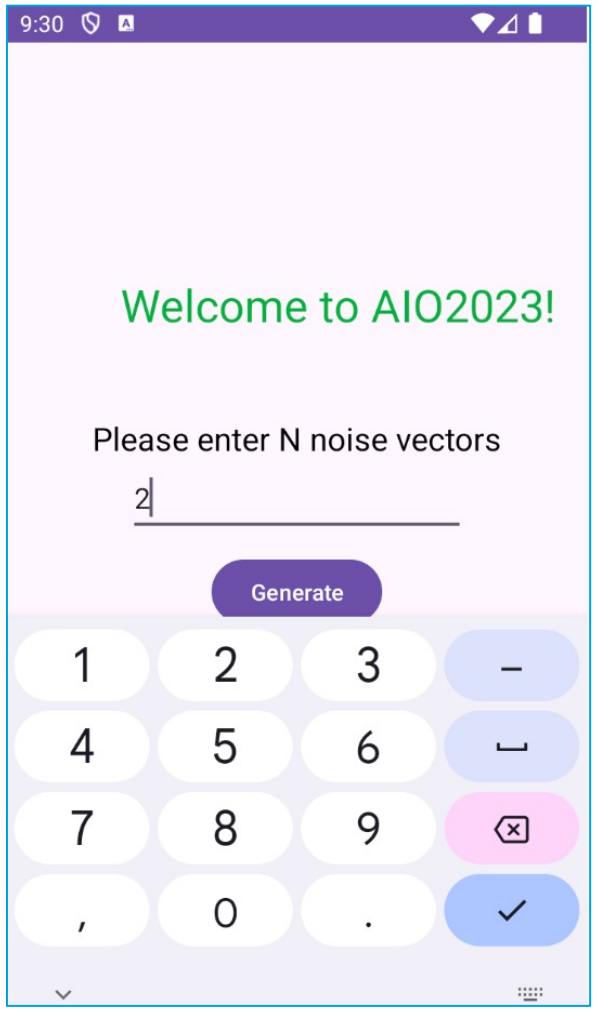
- First is the *Eager mode.* It is built for faster prototyping, training, and experimentation.
- Second is the *Script mode.* It is focused on the production use case. It has 2 components *PyTorch JIT* and *TorchScript.*

```
# Example 2.2 torch.jit.script gpu version

script_cell_gpu = torch.jit.script(model_ft_gpu, (x_ft_gpu))
print("torch.jit.script gpu version", np.mean([timer(script_cell_gpu,x_ft.cuda()) for _ in r
```

Outline

- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

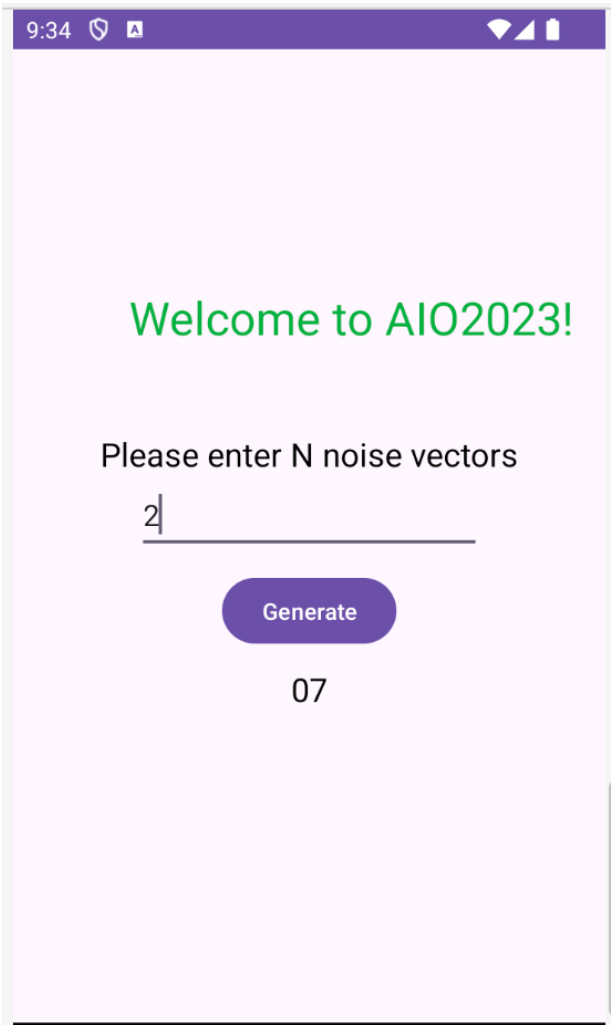
Example 1: Number Generation



Simple model that takes N noise vectors as input and outputs N numbers between 0 and 9

The model will take in a vector with 2 elements (which is just some arbitrary number I chose) and output a vector with 10 elements where each element in the output vector is the probability the model thinks that number should be chosen

Input: 2
Output: 07



Model Creation

```python
import torch
from torch import nn
class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.MLP = nn.Sequential(
            nn.Linear(2, 5),
            nn.Linear(5, 10),
            nn.Linear(10, 15),
            nn.Linear(15, 20),
            nn.Linear(20, 15),
            nn.Linear(15, 10),
            nn.Softmax(-1)
        )

    def forward(self, X):
        return torch.argmax(self.MLP(X), dim=-1)
```

Basic MLP with 2 inputs, 4 hidden layers
and 10 outputs where each output is
the softmax probabilities of a number 0 to 9

Model Evaluation

```
def main():
    # Create the model
    model = Model()

    # Create 4 random noise vectors
    # meaning we want 4 random numbers
    X = torch.distributions.uniform.Uniform(-10000,\
        10000).sample((4, 2))

    # Send the noise vectors through the model
    # to get the argmax outputs
    outputs = model(X)

    # Print the outputs
    for o in outputs:
        print(f"{o.item()} ")

    # Save the model to a file named model.pkl
    torch.save(model.state_dict(), "model.pkl")
```

Basic MLP with 2 inputs, 4 hidden layers
and 10 outputs where each output is
the softmax probabilities of a number 0 to 9

```
main()
```

```
8
8
8
8
1
```

Before working on the app, you can imagine a normal model has several problems. For example:
A good model is probably very large
A model can take a while to load
Models can take a while to make a prediction

Optimize the model for deployment

To reduce these problems, PyTorch has a very useful feature called TorchScript which optimizes the model for deployment

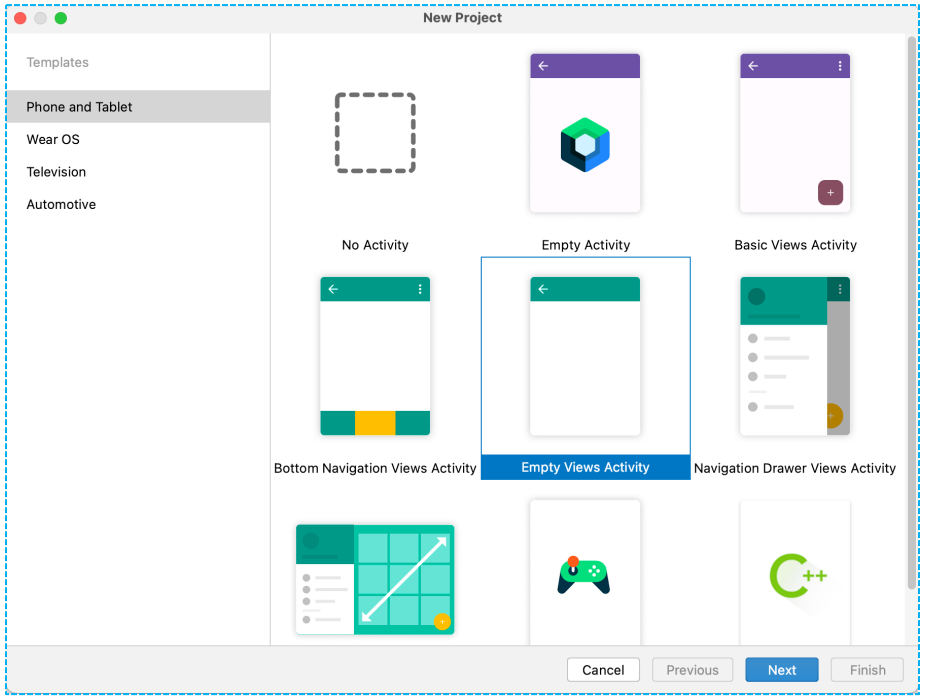
```
from torch.utils.mobile_optimizer import optimize_for_mobile
def optimizeSave():
    # Load in the model
    model = Model()
    model.load_state_dict(torch.load("model.pkl", \
        map_location=torch.device("cpu")))
    model.eval() # Put the model in inference mode

    # Generate some random noise
    X = torch.distributions.uniform.Uniform(-10000, \
        10000).sample((4, 2))

    # Generate the optimized model
    traced_script_module = torch.jit.trace(model, X)
    traced_script_module_optimized = optimize_for_mobile(\
        traced_script_module)

    # Save the optimzied model
    traced_script_module_optimized._save_for_lite_interpreter(\
        "model.pt")
```

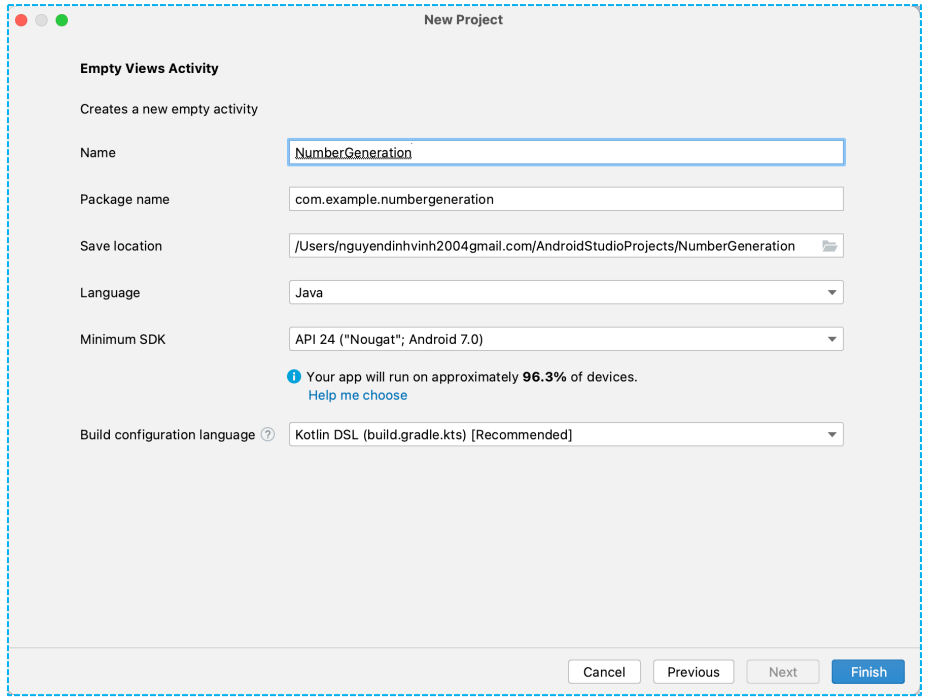
```
import torch
from torch.utils.mobile_optimizer import optimize_for_mobile

torchscript_model = # path to torchscript file
export_model_name = # path to save the .ptl file

model = torch.jit.load(torchscript_model)
optimized_model = optimize_for_mobile(model)
optimized_model._save_for_lite_interpreter(export_model_name)

print(f"mobile optimized model exported to {export_model_name}")
```

Now we have an optimized model to put in our app!

Creating the Mobile App

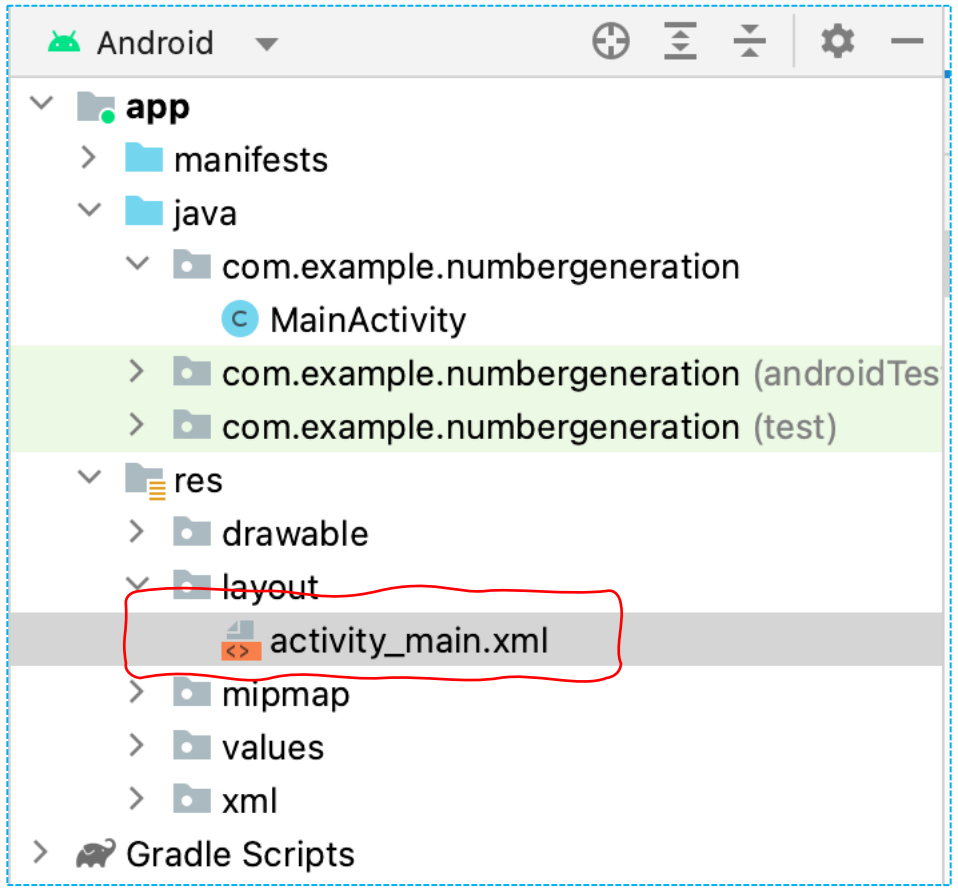
Step 1: Create New Project





Creating the Mobile App

Step 2: Create UI

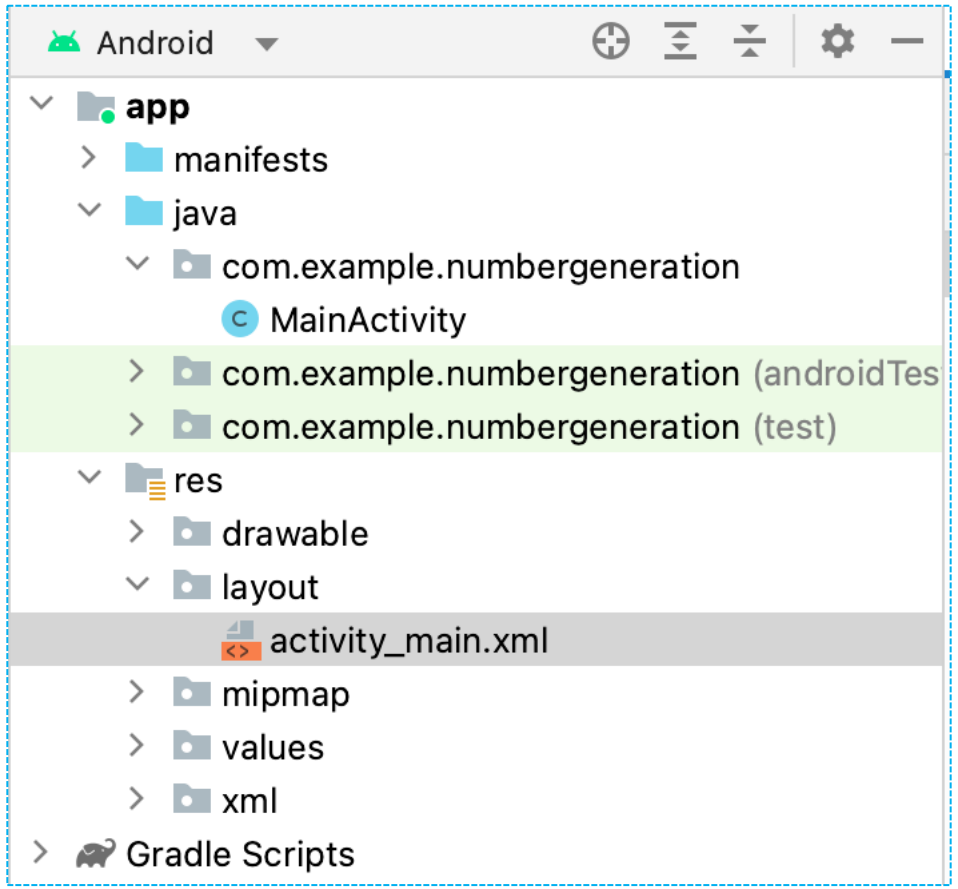


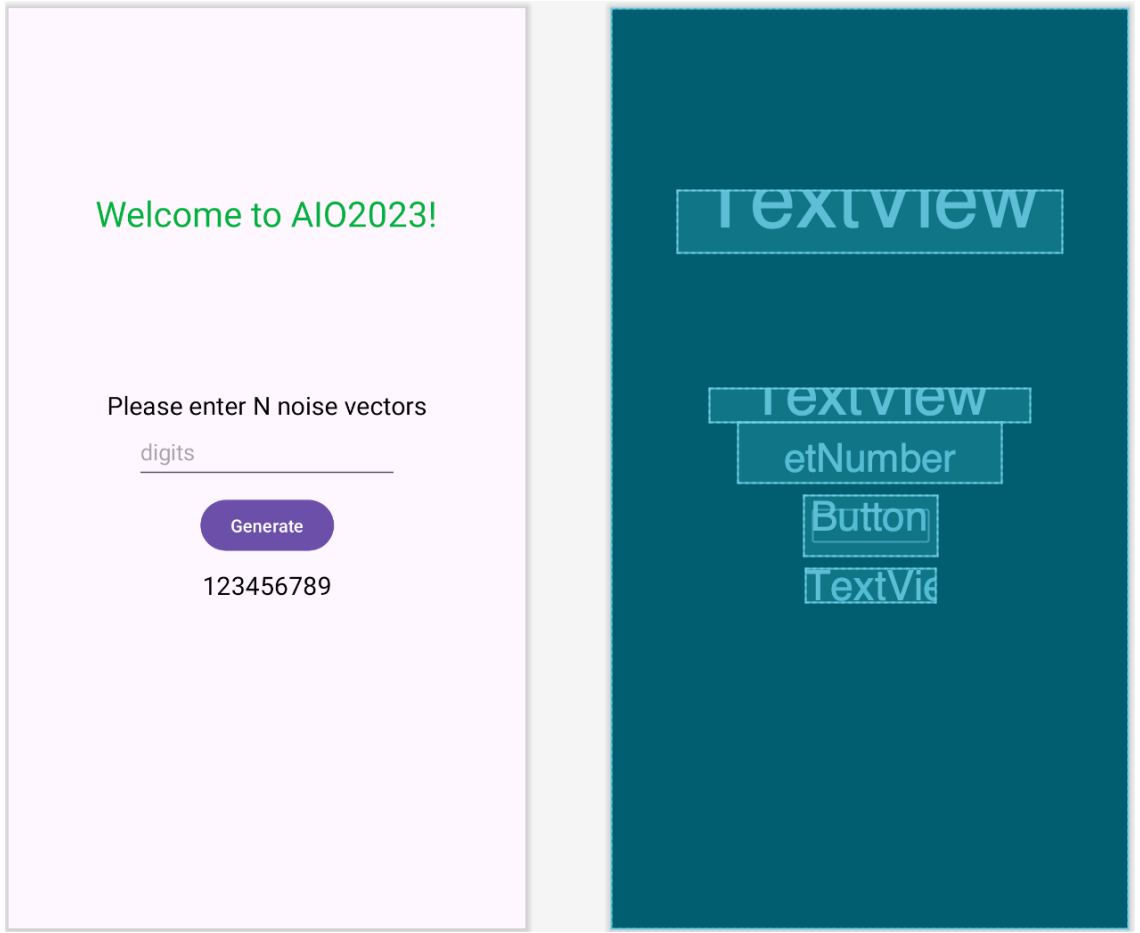
XML Design

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="100dp"
        android:text="Please enter N noise vectors"
        android:textColor="@color/black"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

Creating the Mobile App

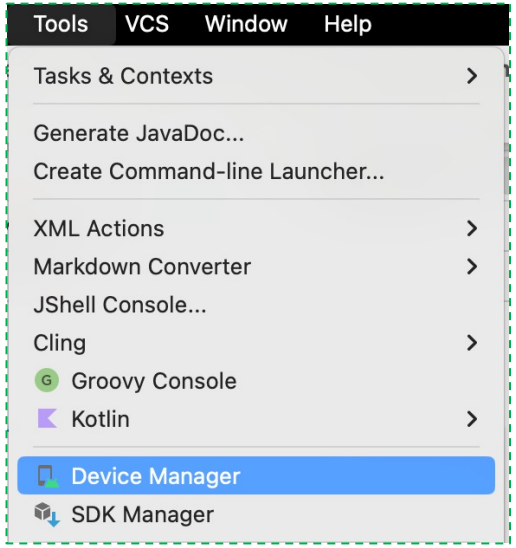
Step 2: Create UI

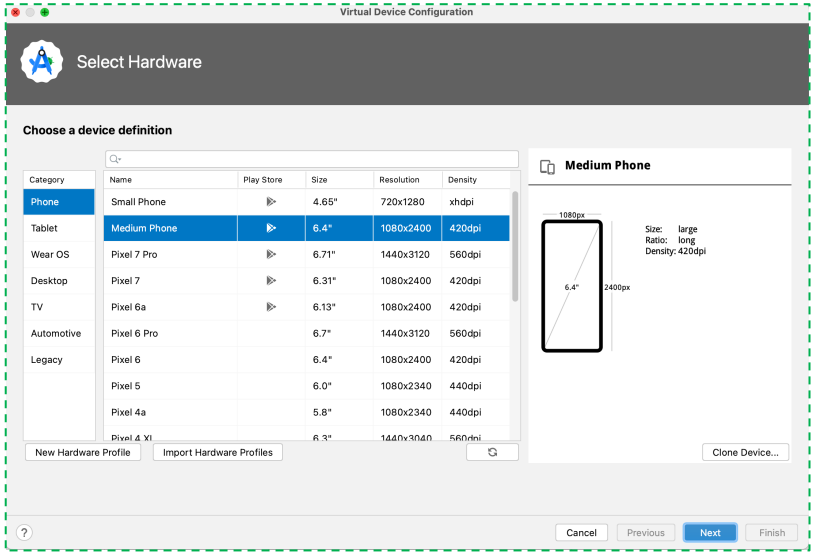


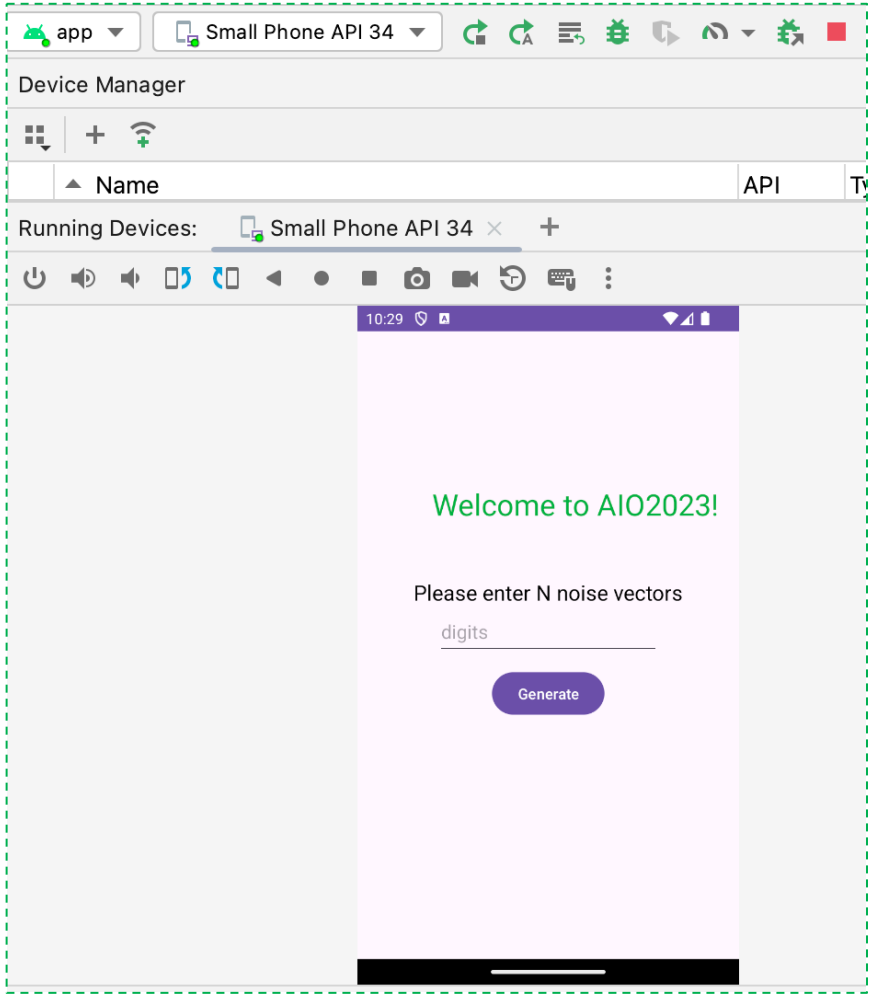


Creating the Mobile App

Step 3: Run apps on the Android Emulator

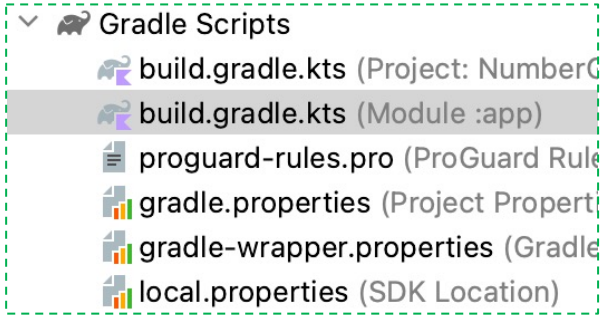






Creating the Mobile App

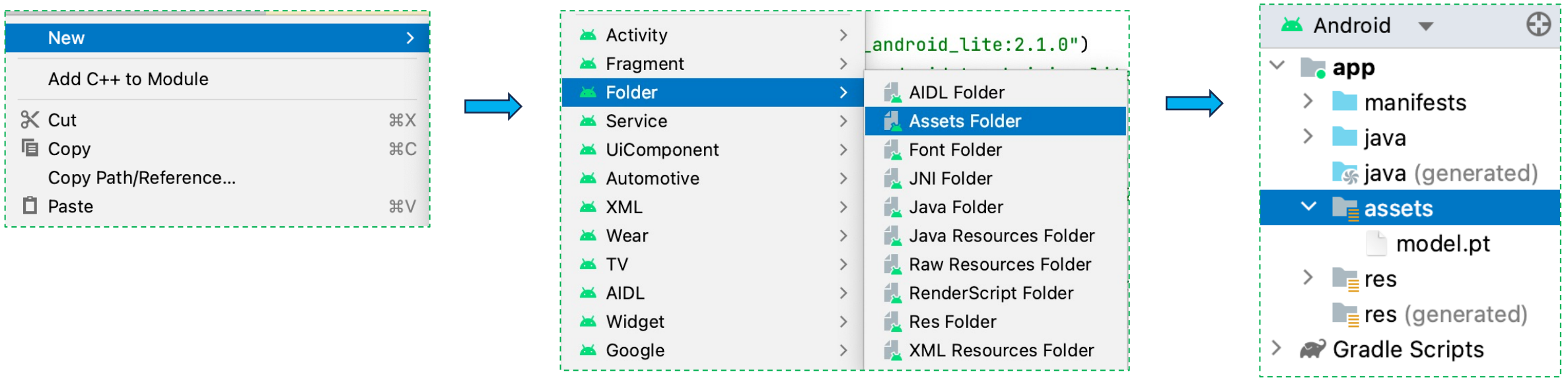
Step 4: PyTorch Android API Configuration



```
build.gradle.kts (:app)
You can use the Project Structure dialog to view and edit your project configuration
21              isMinifyEnabled = false
22              proguardFiles(getDefaultProguardFile( name: "proguard-android-opt
23          }
24      }
25      compileOptions { this: CompileOptions
26          sourceCompatibility = JavaVersion.VERSION_1_8
27          targetCompatibility = JavaVersion.VERSION_1_8
28      }
29  }
30
31  dependencies { this: DependencyHandlerScope
32
33      implementation("org.pytorch:pytorch_android_lite:2.1.0")
34      implementation("org.pytorch:pytorch_android_torchvision_lite:2.1.0")
35
36      implementation("androidx.appcompat:appcompat:1.6.1")
37      implementation("com.google.android.material:material:1.11.0")
38      implementation("androidx.constraintlayout:constraintlayout:2.1.4")
39      testImplementation("junit:junit:4.13.2")
40      androidTestImplementation("androidx.test.ext:junit:1.1.5")
41      androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
```

Creating the Mobile App

Step 5: Create Assests folder to store Pytorch Lite Model



Creating the Mobile App

Step 6: Update MainActivtity to use Pytorch Android Library

MainActivity.java

```
21
22  public class MainActivity extends AppCompatActivity {
23      // Elements in the view
        3 usages
24      EditText etNumber;
        2 usages
25      Button btnInfer;
        2 usages
26      TextView tvDigits;
27
28      // Tag used for logging
        1 usage
29      private static final String TAG = "MainActivity";
30
31      // PyTorch model
        2 usages
32      Module module;
33
34
35      // Given the name of the pytorch model, get the path for that model
        1 usage
36      public String assetFilePath(String assetName) throws IOException {
37          File file = new File(this.getFilesDir(), assetName);
38          if (file.exists() && file.length() > 0) {
39              return file.getAbsolutePath();
40          }
```

```
public Tensor generateTensor(long[] Size) {
    // Create a random array of floats
    Random rand = new Random();
    float[] arr = new float[(int)(Size[0]*Size[1])];
    for (int i = 0; i < Size[0]*Size[1]; i++) {
        arr[i] = -10000 + rand.nextFloat() * (20000);
    }

    // Create the tensor and return it
    return Tensor.fromBlob(arr, Size);
}
```

```
// Load in the model
        try {
            module = LiteModuleLoader.load(assetFilePath( assetName: "model.pt"));
        } catch (IOException e) {
            Log.e(TAG,  msg: "Unable to load model", e);
        }
```

Number Generation: Demo

Outline

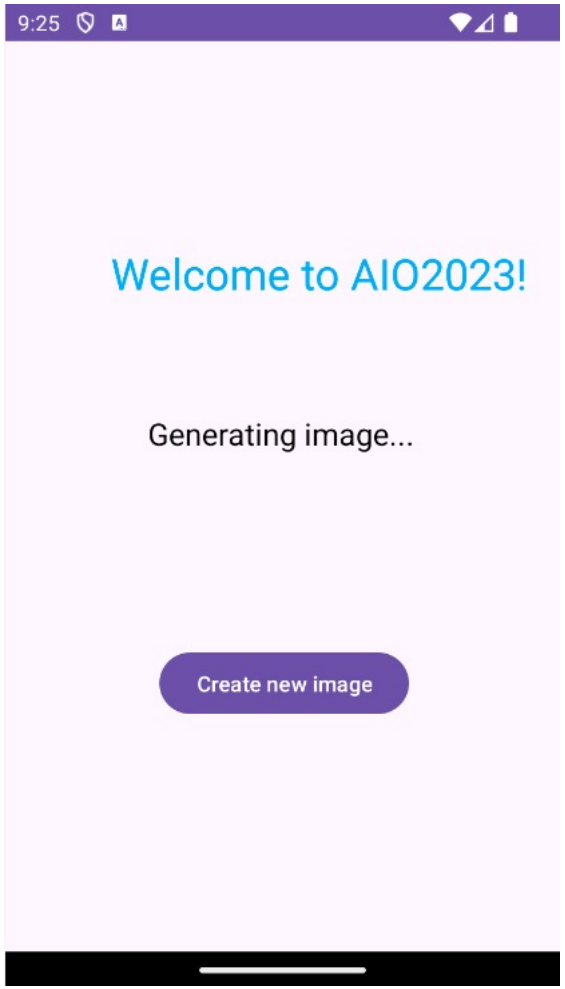
- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

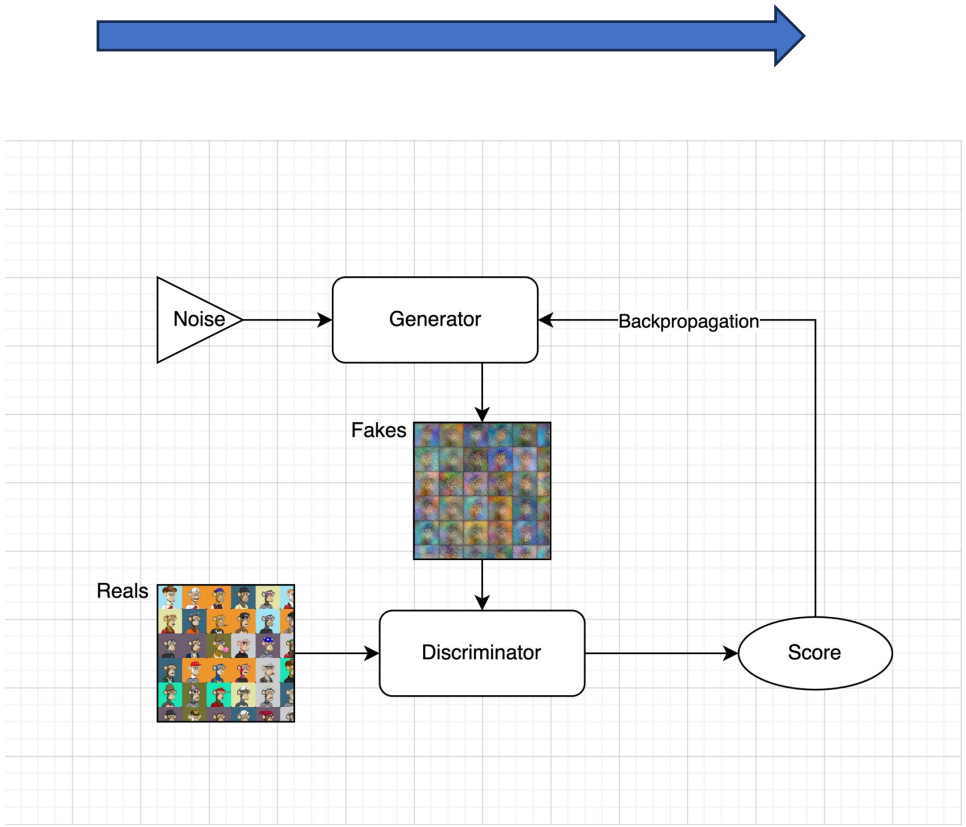
Example 2: Image Generation

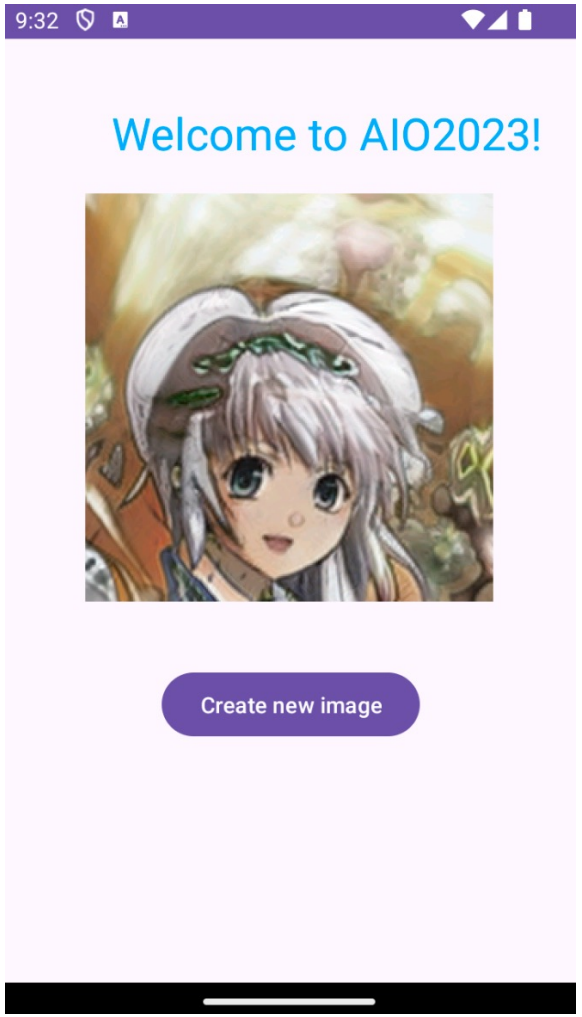
StyleGAN3 on Anime Faces

https://blog.runpod.io/training-stylegan3-on-runpod/

Example 2: Image Generation





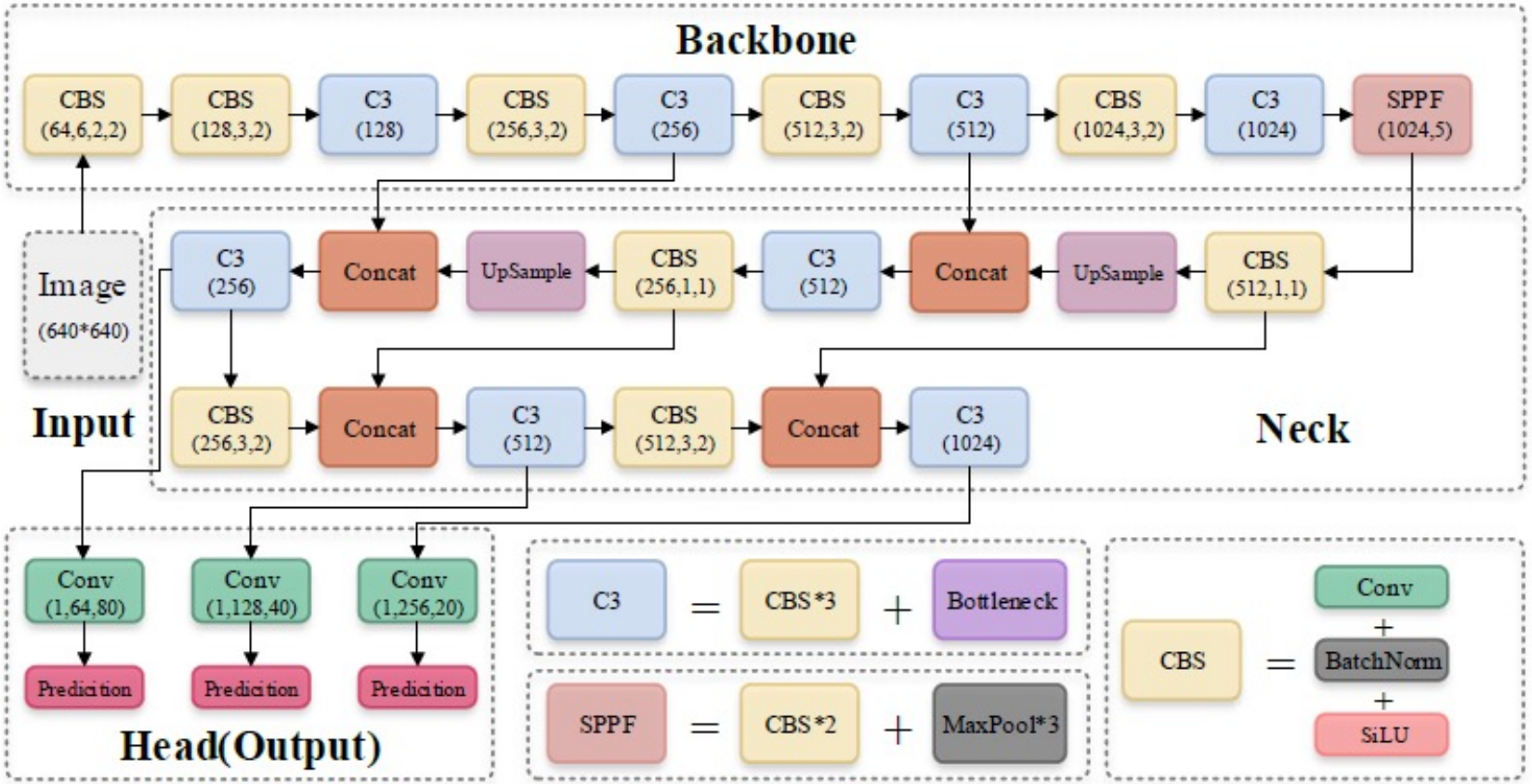


Outline

- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

Example 3: Object Detection with YOLO

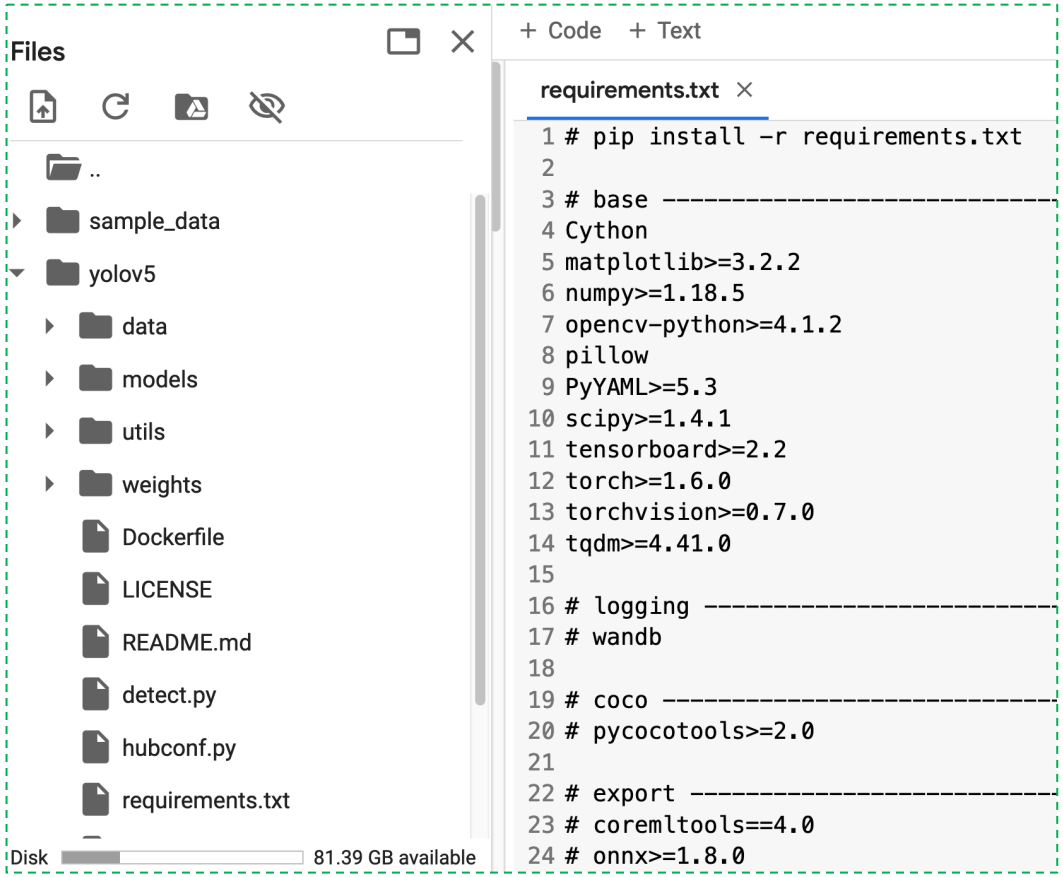
YOLO (You Only Look Once) is one of the fastest and most popular object detection models. YOLOv5 is an open-source implementation of the latest version of YOLO. This Object Detection with YOLOv5 Android sample app uses the PyTorch scripted YOLOv5 model to detect objects of the 80 classes trained with the model.



Example 3: Object Detection with YOLO

Step 1: Install Yolov5

```
!git clone https://github.com/jeffxtang/yolov5
%cd yolov5
!pip install -r /content/yolov5/requirements.txt
```

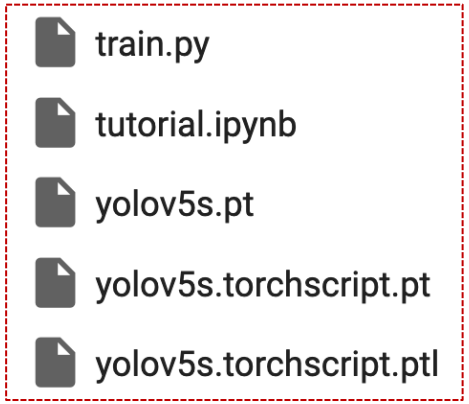


Example 3: Object Detection with YOLO

Step 2: Generate the optimized TorchScript lite model yolov5s.torchscript.ptl

```
!python models/export.py
```

```
Namespace(weights='./yolov5s.pt', img_size=[640, 640], batch_size=1)
Downloading https://github.com/ultralytics/yolov5/releases/download/v3.1/yolo
100% 14.5M/14.5M [00:00<00:00, 197MB/s]

Fusing layers...
Model Summary: 140 layers, 7.45958e+06 parameters, 0 gradients
/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning:
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]

Starting TorchScript export with torch 2.1.0+cu121...
/content/yolov5/./models/yolo.py:47: TracerWarning: Conver
  if self.grid[i].shape[2:4] != x[i].shape[2:4]:
/usr/local/lib/python3.10/dist-packages/torch/jit/_trace.p
  module._c._create_method_from_trace(
TorchScript export success, saved as ./yolov5s.torchscript
```

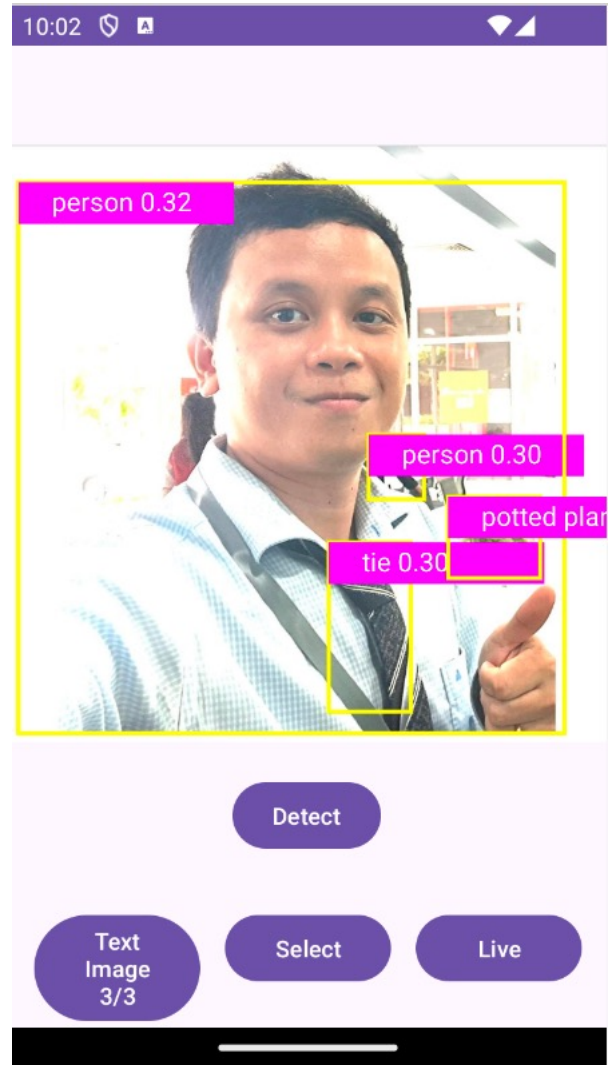



```python
# TorchScript export
try:
    print('\nStarting TorchScript export with torch %s...' % torch.__version__)
    jit_fn = opt.weights.replace('.pt', '.torchscript.pt')
    lite_fn = opt.weights.replace('.pt', '.torchscript.ptl')
    ts = torch.jit.trace(model, img)
    from torch.utils.mobile_optimizer import optimize_for_mobile
    ts = optimize_for_mobile(ts)
    ts.save(jit_fn)
    ts._save_for_lite_interpreter(lite_fn)
    print('TorchScript export success, saved as %s' % lite_fn)
except Exception as e:
    print('TorchScript export failure: %s' % e)
```

Example 3: Object Detection with YOLO

Step 3: Design Android UI

```
<androidx.constraintlayout.widget.ConstraintL
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/black"
    android:fitsSystemWindows="true">

    <ViewStub
        android:id="@+id/object_detection_texture_view_
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout="@layout/texture_view"
        app:layout_constraintBottom_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.example.yolo_mobile.ResultView
        android:id="@+id/resultView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```



Example 3: Object Detection with YOLO

Step 4: Load Yolov5 model and class label

```
try {
    mModule = LiteModuleLoader.load(MainActivity.assetFilePath(getApplicationContext(), assetName: "yolov5s.torchscript.ptl"));
    BufferedReader br = new BufferedReader(new InputStreamReader(getAssets().open( fileName: "classes.txt")));
    String line;
    List<String> classes = new ArrayList<>();
    while ((line = br.readLine()) != null) {
        classes.add(line);
    }
    PrePostProcessor.mClasses = new String[classes.size()];
    classes.toArray(PrePostProcessor.mClasses);
} catch (IOException e) {
    Log.e( tag: "Object Detection", msg: "Error reading assets", e);
    finish();
}
```

Example 3: Object Detection with YOLO

Step 5: Use Yolov5 for detection

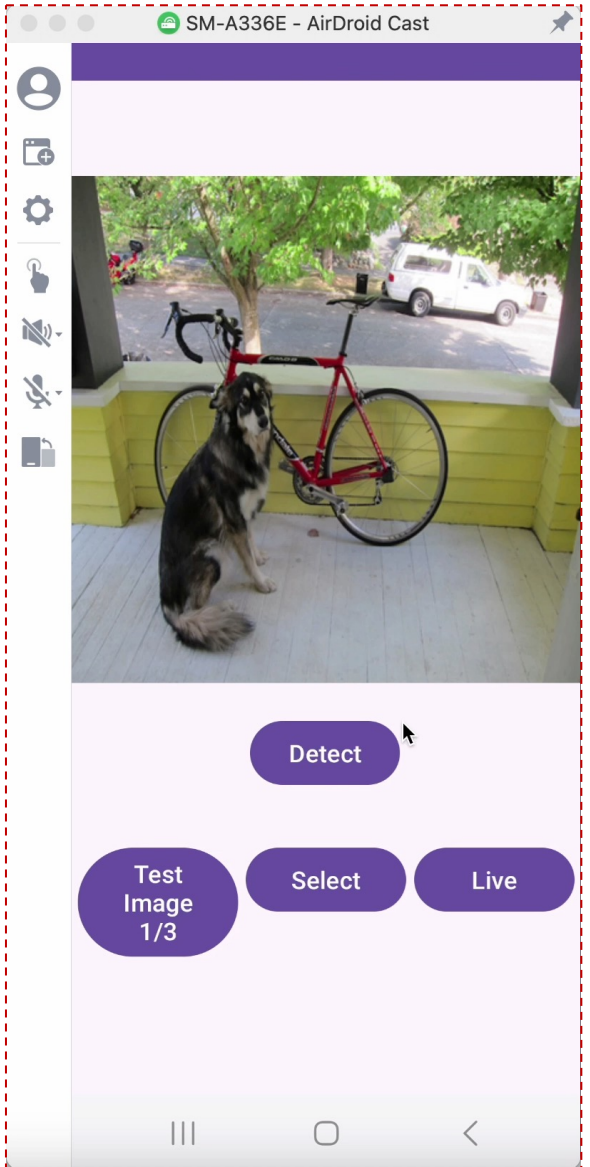
```
Bitmap bitmap = imgToBitmap(image.getImage());
Matrix matrix = new Matrix();
matrix.postRotate( degrees: 90.0f);
bitmap = Bitmap.createBitmap(bitmap,  x: 0,  y: 0, bitmap.getWidth(), bitmap.getHeight(), matrix,  filter: true);
Bitmap resizedBitmap = Bitmap.createScaledBitmap(bitmap, PrePostProcessor.mInputWidth, PrePostProcessor.mInputHeight,  filter: true)

final Tensor inputTensor = TensorImageUtils.bitmapToFloat32Tensor(resizedBitmap, PrePostProcessor.NO_MEAN_RGB, PrePostProcessor.
IValue[] outputTuple = mModule.forward(IValue.from(inputTensor)).toTuple();
final Tensor outputTensor = outputTuple[0].toTensor();
final float[] outputs = outputTensor.getDataAsFloatArray();

float imgScaleX = (float)bitmap.getWidth() / PrePostProcessor.mInputWidth;
float imgScaleY = (float)bitmap.getHeight() / PrePostProcessor.mInputHeight;
float ivScaleX = (float)mResultView.getWidth() / bitmap.getWidth();
float ivScaleY = (float)mResultView.getHeight() / bitmap.getHeight();
```

Example 3: Object Detection with YOLO

Step 6: Setup CameraX for live detection

```
private void setupCameraX() {
    final TextureView textureView = getCameraPreviewTextureView();
    final PreviewConfig previewConfig = new PreviewConfig.Builder().build();
    final Preview preview = new Preview(previewConfig);
    preview.setOnPreviewOutputUpdateListener(output -> textureView.setSurfaceTexture(output.getSurfaceTexture()));

    final ImageAnalysisConfig imageAnalysisConfig =
        new ImageAnalysisConfig.Builder()
            .setTargetResolution(new Size( width: 480,  height: 640))
            .setCallbackHandler(mBackgroundHandler)
            .setImageReaderMode(ImageAnalysis.ImageReaderMode.ACQUIRE_LATEST_IMAGE)
            .build();
    final ImageAnalysis imageAnalysis = new ImageAnalysis(imageAnalysisConfig);
    imageAnalysis.setAnalyzer((image, rotationDegrees) -> {
        if (SystemClock.elapsedRealtime() - mLastAnalysisResultTime < 500) {
            return;
        }

        final R result = analyzeImage(image, rotationDegrees);
        if (result != null) {
            mLastAnalysisResultTime = SystemClock.elapsedRealtime();
            runOnUiThread(() -> applyToUiAnalyzeImageResult(result));
        }
    });

    CameraX.bindToLifecycle( lifecycleOwner: this, preview, imageAnalysis);
}
```

Example 3: Object Detection with YOLO




TIME FOR
A REFRESH

Mini Quiz

Kahoot!

Mobile Application Development Using ML

1 play · 1 player

Start | Assign | Play solo

A public kahoot

aivietnameduvn
Updated 38 seconds ago

Questions (4) Show answers

1 - Quiz
What is PyTorch JIT's benefits? PyTorch JIT is an optimized compiler for PyTorch programs.


2 - Quiz
How can we deploy ML algorithm on Mobile apps?
20 sec

3 - Quiz
What is TorchScript's support? TorchScript is a static high-performance subset of Python language
20 sec

4 - Quiz
Why is Machine Learning Deployment Hard?
20 sec

Resource credits

Outline

- Deployment Workflow of Machine Learning in Mobile
- Example 1: Number Generation
- Example 2: Image Generation
- Example 3: Object Detection with Yolo
- Example 4: Image Classification using Transfer Learning

Example 4: Transfer Learning

Train a convolutional neural network for image classification using transfer learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

These two major transfer learning scenarios look as follows:

- **Finetuning the ConvNet**: Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on imagenet 1000 dataset. Rest of the training looks as usual.
- **ConvNet as fixed feature extractor**: Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

Example 4: Transfer Learning

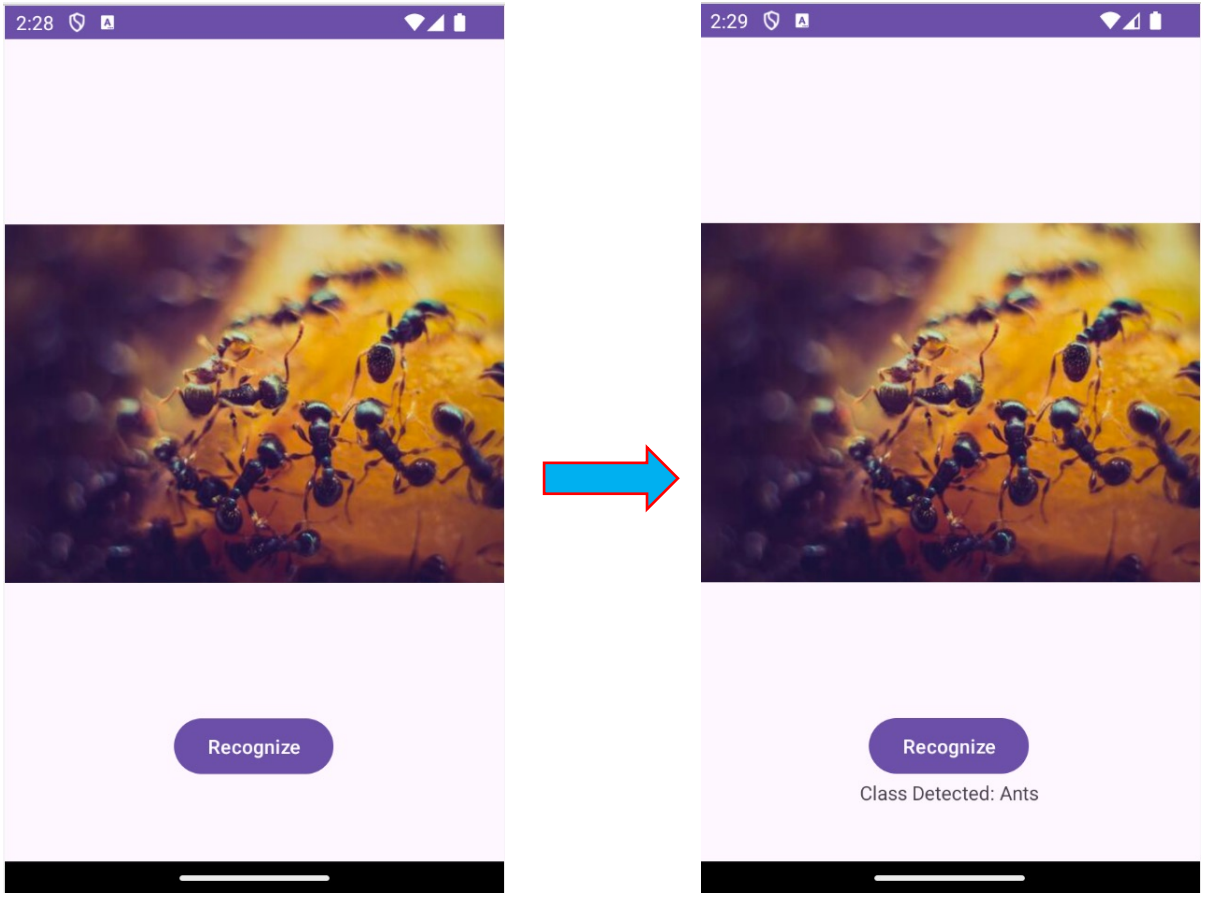
Train a convolutional neural network for image classification using transfer learning

The problem we're going to solve today is to train a model to classify **ants** and **bees**. We have about 120 training images each for ants and bees. There are 75 validation images for each class. Usually, this is a very small dataset to generalize upon, if trained from scratch. Since we are using transfer learning, we should be able to generalize reasonably well.

Example 4: Transfer Learning

Ants vs Bee Classification

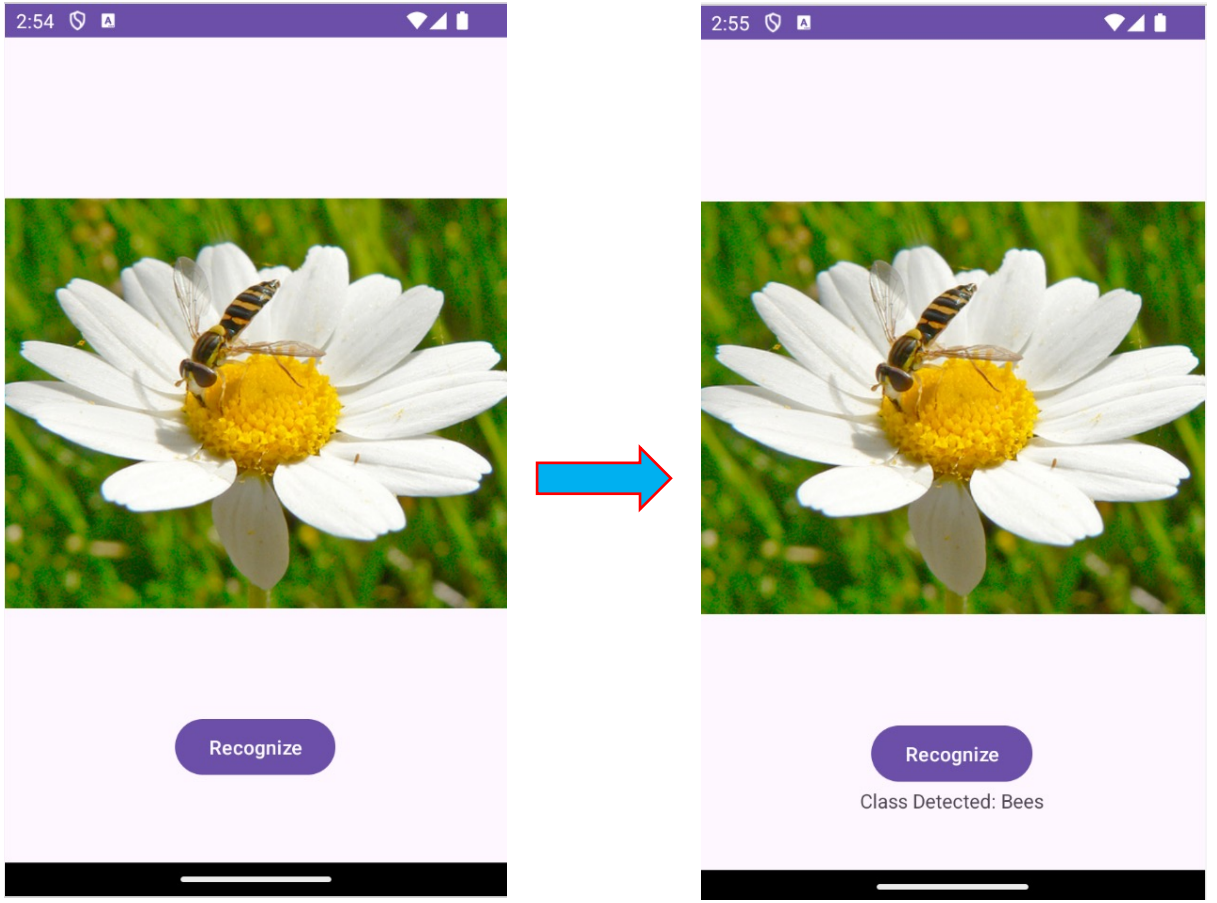


Example 4: Transfer Learning

Ants vs Bee Classification



Example 4: Transfer Learning

Step 1: Dataset Preparation

Ants vs Bee Classification




```
data_dir = '/content/hymenoptera_data'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
                                          data_transforms[x])
                  for x in ['train', 'val']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size=4,
                                              shuffle=True, num_workers=4)
               for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes
```

Example 4: Transfer Learning

Ants vs Bee Classification

Step 2:

- Data augmentation and normalization for training
- Normalization for validation

```
data_transforms = {
    'train': transforms.Compose([
        transforms.ToTensor(),
        transforms.RandomResizedCrop(224, antialias=True),
        transforms.RandomHorizontalFlip(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.ToTensor(),
        transforms.Resize(256, antialias=True),
        transforms.CenterCrop(224),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

Example 4: Transfer Learning

Step 3:

- Train the model

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()

    # Create a temporary directory to save training checkpoints
    with TemporaryDirectory() as tempdir:
        best_model_params_path = os.path.join(tempdir, 'best_model_params.pt')

        torch.save(model.state_dict(), best_model_params_path)
        best_acc = 0.0

        for epoch in range(num_epochs):
            print(f'Epoch {epoch}/{num_epochs - 1}')
            print('-' * 10)

            # Each epoch has a training and validation phase
            for phase in ['train', 'val']:
                if phase == 'train':
                    model.train()  # Set model to training mode
                else:
                    model.eval()   # Set model to evaluate mode

                running_loss = 0.0
                running_corrects = 0
```

Example 4: Transfer Learning

Step 4:

- Finetuning the ConvNet
- Load a pretrained model and reset final fully connected layer

```
model_ft = models.resnet18(weights='IMAGENET1K_V1')
num_ftrs = model_ft.fc.in_features
# 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_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that all parameters are being optimized
optimizer_ft = optim.SGD(model_ft.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_ft, step_size=7, gamma=0.1)
```

Example 4: Transfer Learning

Step 5:

- Create a class to run an inference over an image using our Pytorch model

```
class my_Model(torch.nn.Module):
    def __init__(self, model_path=None):
        super().__init__()

        #Building the model
        self.model = torchvision.models.resnet18(weights='IMAGENET1K_V1')
        for param in self.model.parameters():
            param.requires_grad = False

        num_ftrs = self.model.fc.in_features
        self.model.fc = nn.Linear(num_ftrs, 2)

        weights = torch.load(model_path, map_location=torch.device('cpu'))
        self.model.load_state_dict(weights)
        self.model.to('cpu')
        self.model.eval()

        #We need to make the transformations sequential
        self.transforms = torch.nn.Sequential(
            transforms.Resize([256], antialias=True),
            transforms.CenterCrop(224),
            transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
        )
```

Example 4: Transfer Learning

Step 6:

❑ Converting to TorchScript

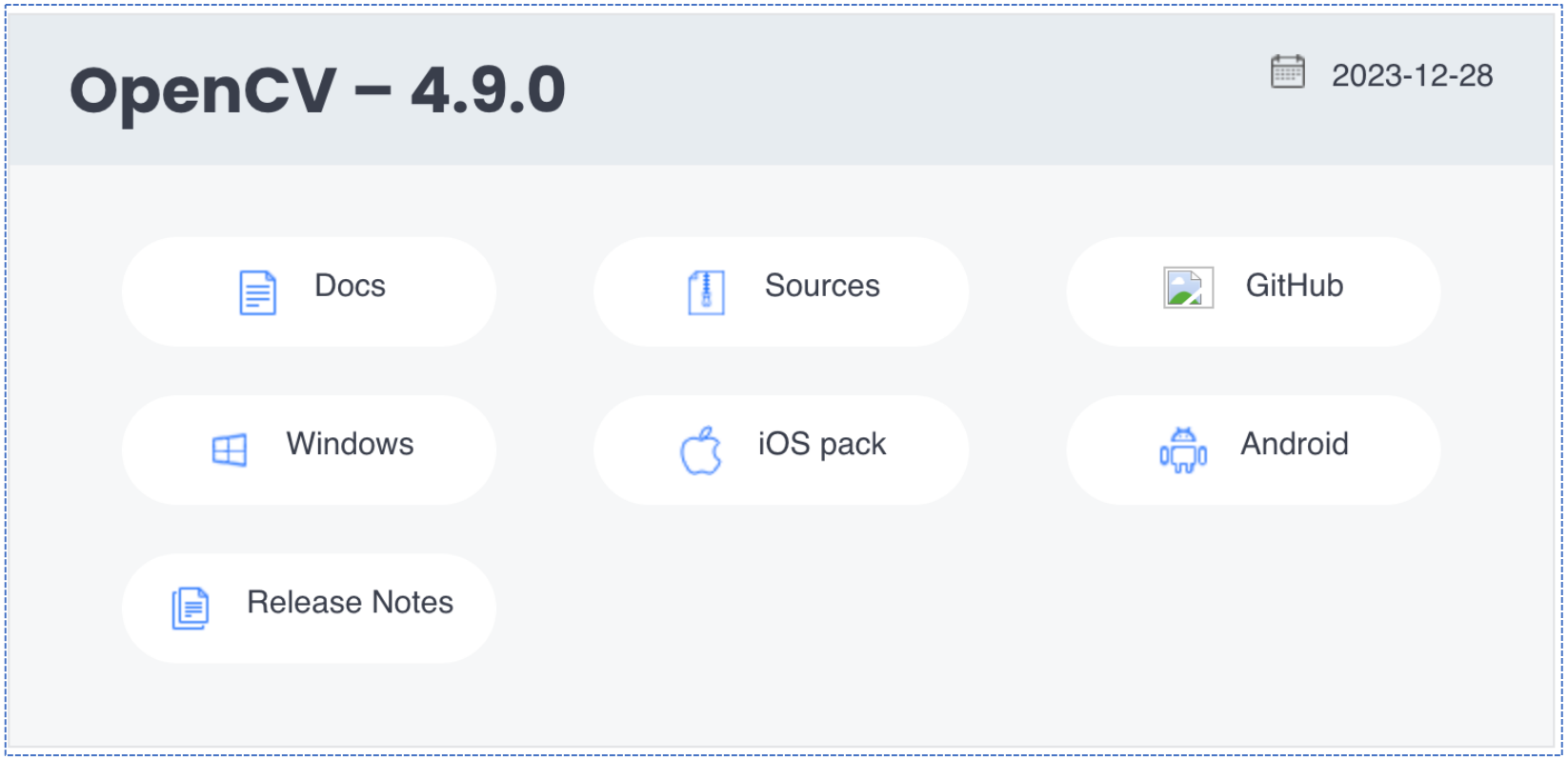
Pytorch has multiple operators and methods that are not supported by Android. For that purpose, Pytorch Mobile was created. It enables us to convert a Pytorch model to a model that our mobile application is able to work with

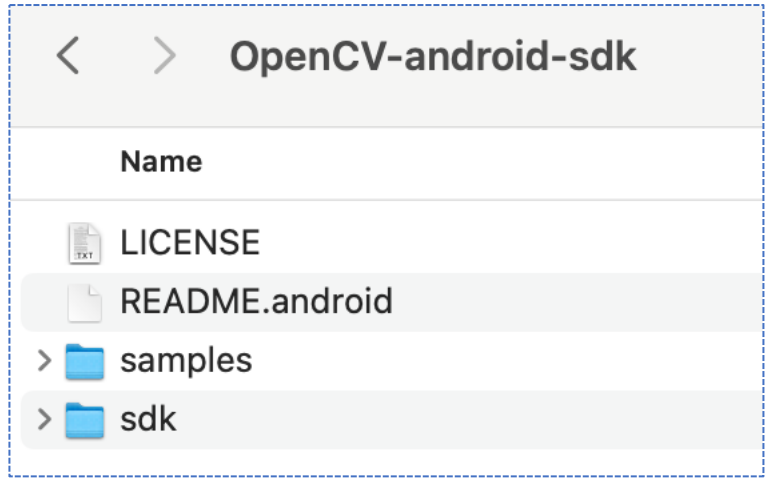
```
from torch.utils.mobile_optimizer import optimize_for_mobile

scripted_module = torch.jit.script(my_model)
optimized_scripted_module = optimize_for_mobile(scripted_module)
# using optimized lite interpreter model makes inference about 60% faster
optimized_scripted_module._save_for_lite_interpreter("my_model_lite.ptl")
```

Example 4: Transfer Learning

Step 7:

- Import the Pytorch Android and OpenCV dependencies



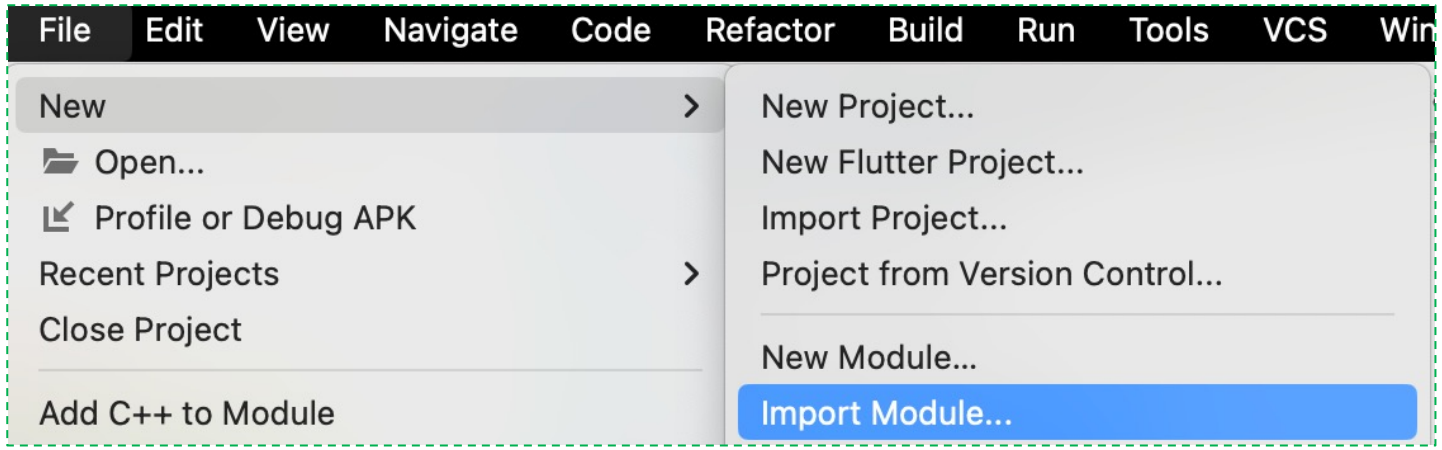


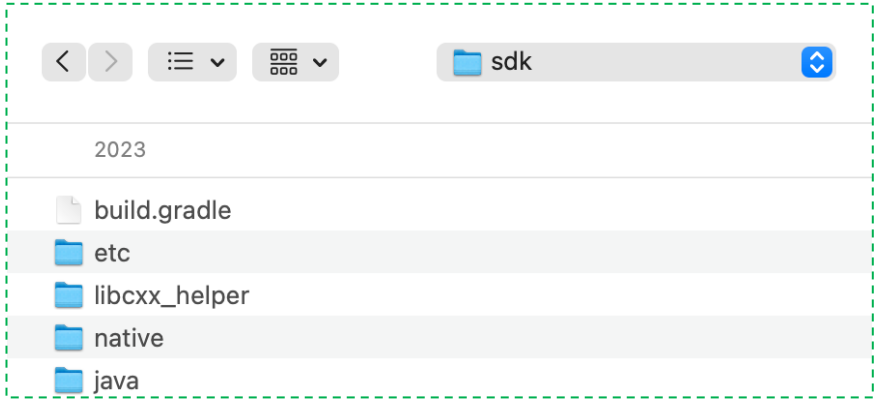
https://opencv.org/releases/

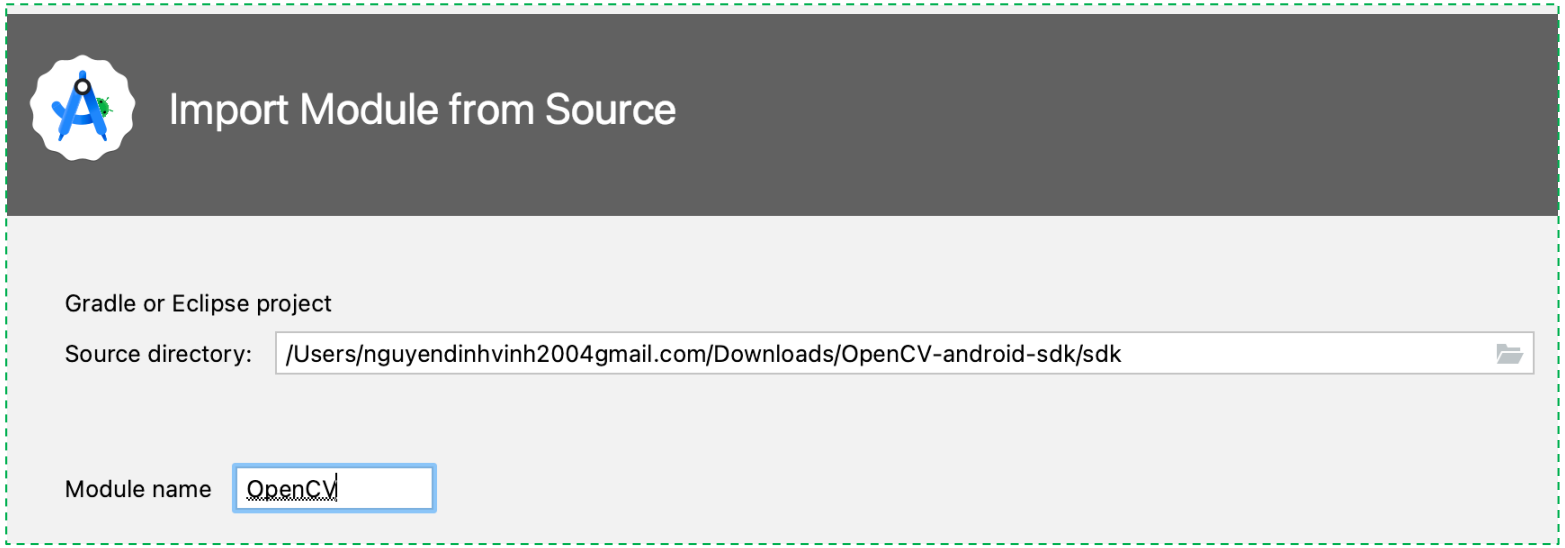
Example 4: Transfer Learning

Step 7:

- Import the Pytorch Android and OpenCV dependencies







Example 4: Transfer Learning

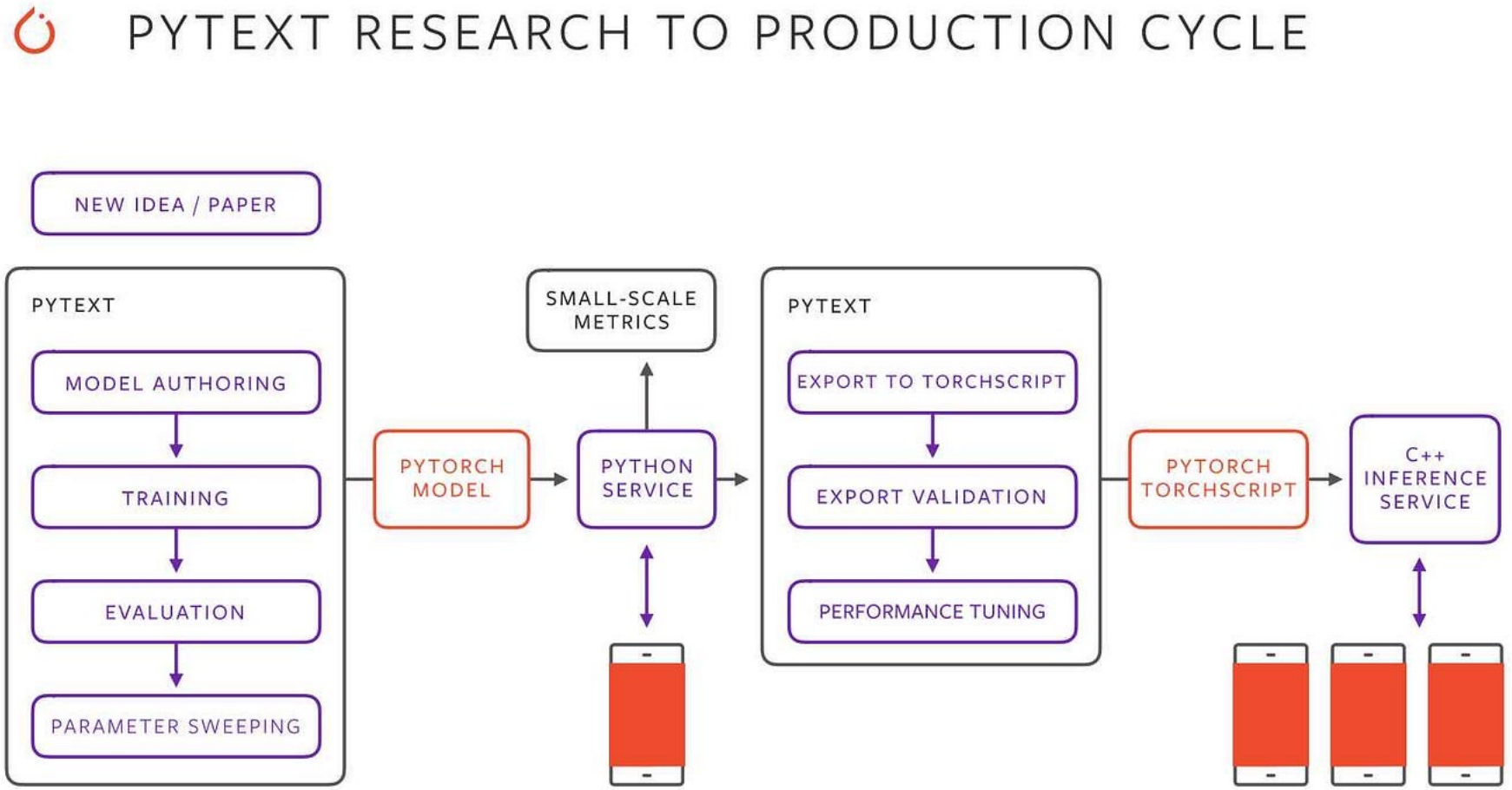
Step 7:

- Import the Pytorch Android and OpenCV dependencies

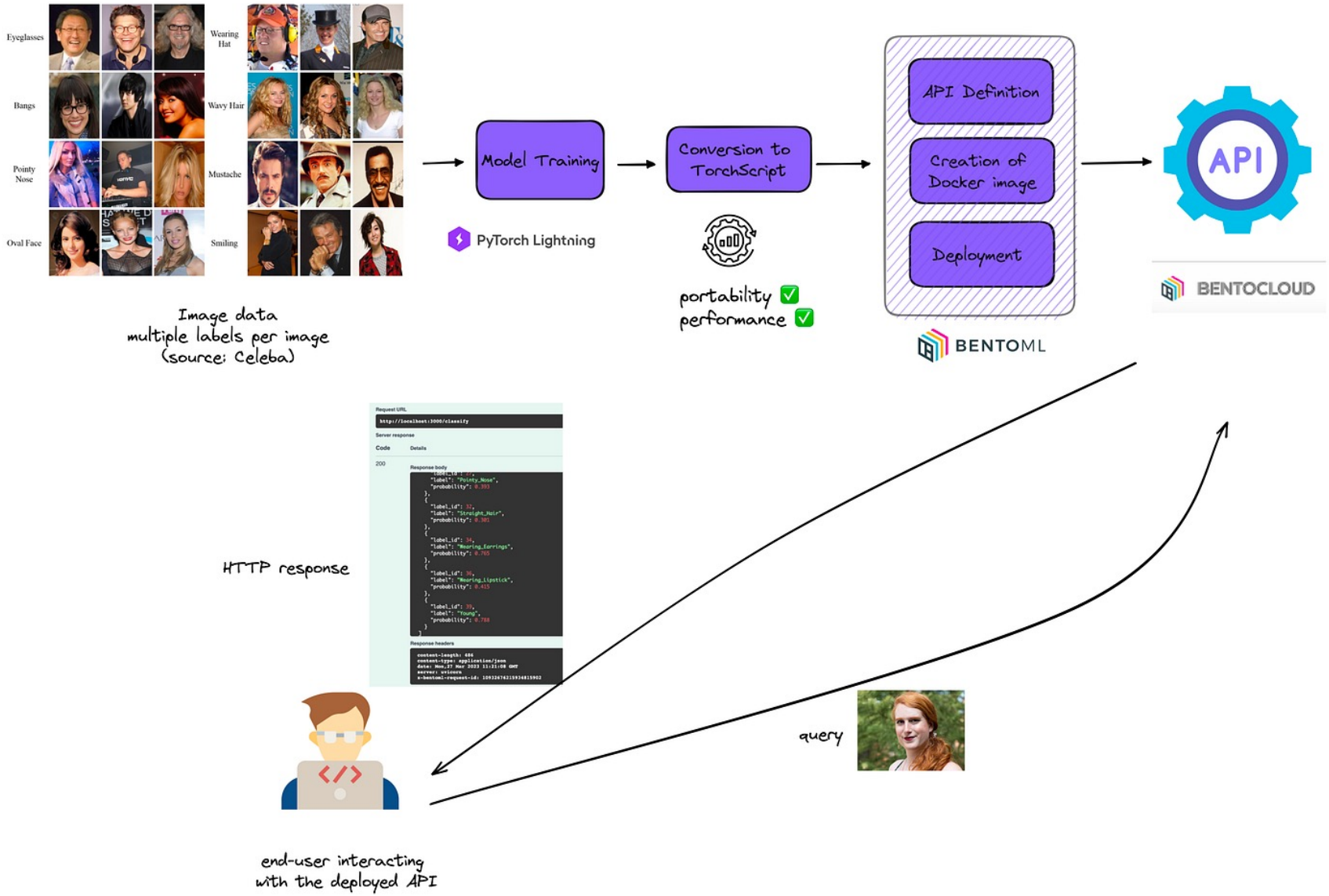
```
dependencies {  this: DependencyHandlerScope
    implementation("org.pytorch:pytorch_android_lite:2.1.0")
    implementation("org.pytorch:pytorch_android_torchvision_lite:2.1.0")
    implementation ("com.quickbirdstudios:opencv:4.5.3.0")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.11.0")
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
}
```

```
packagingOptions {  this: Packaging
    jniLibs.pickFirsts += "**/libc++_shared.so"
}
```

Summary


PYTEXT RESEARCH TO PRODUCTION CYCLE
NEW IDEA / PAPER
PYTEXT
MODEL AUTHORING
TRAINING
EVALUATION
PARAMETER SWEEPING
PYTORCH MODEL
SMALL-SCALE METRICS
PYTHON SERVICE
PYTEXT
EXPORT TO TORCHSCRIPT
EXPORT VALIDATION
PERFORMANCE TUNING
PYTORCH TORCHSCRIPT
C++ INFERENCE SERVICE

Summary


Eyeglasses
Wearing Hat
Bangs
Wavy Hair
Pointy Nose
Mustache
Oval Face
Smiling
Image data
multiple labels per image
(source: Celeba)
Model Training
PyTorch Lightning
Conversion to TorchScript
portability ✅
performance ✅
API Definition
Creation of Docker image
Deployment
BENTOML
API
BENTOCLOUD
HTTP response
query
end-user interacting
with the deployed API


Thank you


Thank you