



# TensorFlow

**Tensorflow Kütüphanesini Kullanarak Model Eğitimi**

**Hazırlayan: Mert Kışlakçı**

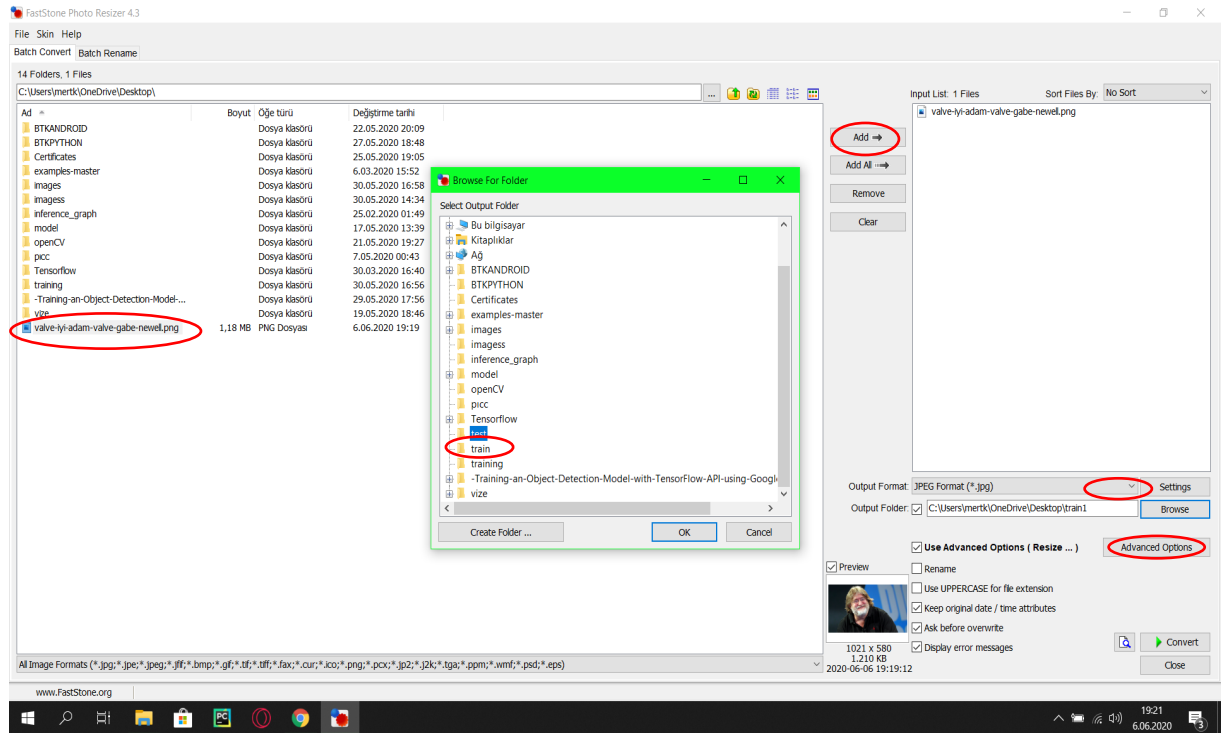
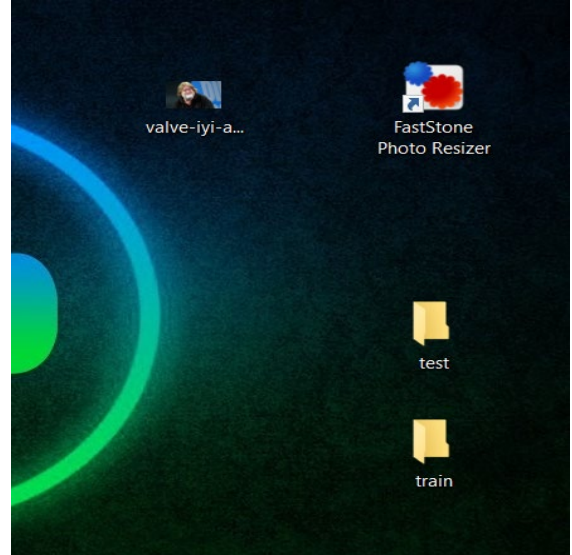
## Datasheet Toplanması

Bu kısımda eğitilmesini istediğimiz modelin görsellerini toplayarak kendi datashettimizi oluşturacağız bunun için öncelikle bu link üzerinden “FastStone” programını indirmemiz gerekiyor.

<https://www.tamindir.com/windows/faststone-photo-resizer/>

Programın kurulumunu yaptıktan sonra yandaki görselde de görüleceği üzere masaüstüne iki adet test ve train klasörü oluşturuyoruz.

Ardından topladığımız resimlerin %20 si test %80 i ise train klasörüne olacak şekilde ayarlıyoruz.

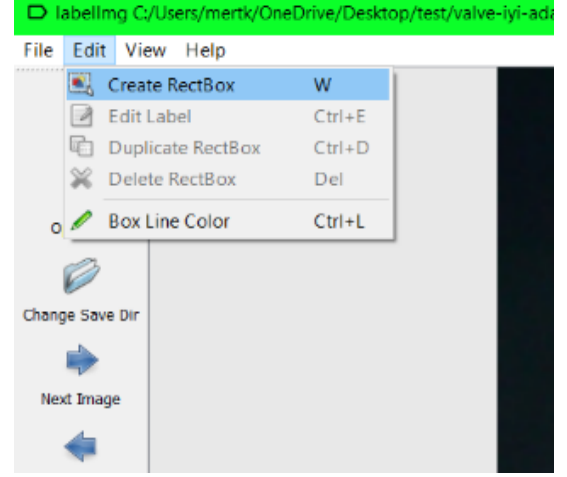


Kullanılan resimlerin .jpg formatında ve de 300x300 pixel boyutunda olması gerekmektedir. Bu bağlamda program üzerinden “Add” komutu ile görsel veya görseller sağ kısma geçirilir ardından “Output Format” olarak .jpg uzantısı seçilir ve daha sonra “Advanced Options” kısmından resmin pixel boyutu 300x300 olarak belirlenir.

## Datasheet XML Formata Dönüştürülmesi

“LabelIMG” programı resimlerin .xml uzantısını çıkartmak için kullanılır. Bu link üzerinden LabelIMG kurulumu gerçekleştirilir ardından test ve train klasörlerindeki resimler tek tek açılarak burada etiketleme yapılır. “Edit” kısmından “Create RectBox” seçilerek etiketleme yapılır. Klavyenin “W” tuşu ile de etiketleme yapabilirsiniz.

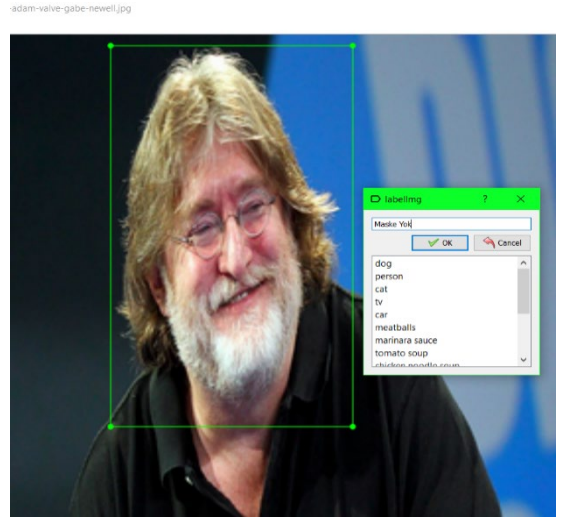
<https://github.com/tzutalin/labelImg>



Benim eğitmeye çalıştığım model insan yüzünde maske olup olmadığını algılayan bir model eğitmek.

Bu bağlamda yüze etiketleme yaparak (gereksiz alanları etiketlemekten olabildiğince kaçının) ardından çıkan etikete “Maske Yok” yazıyorum. Bir görsel üzerinde birden fazla etiketleme yapabilirsiniz. Etiketleme işlemi bittikten sonra aynı klasöre .xml formatını kaydediyoruz.

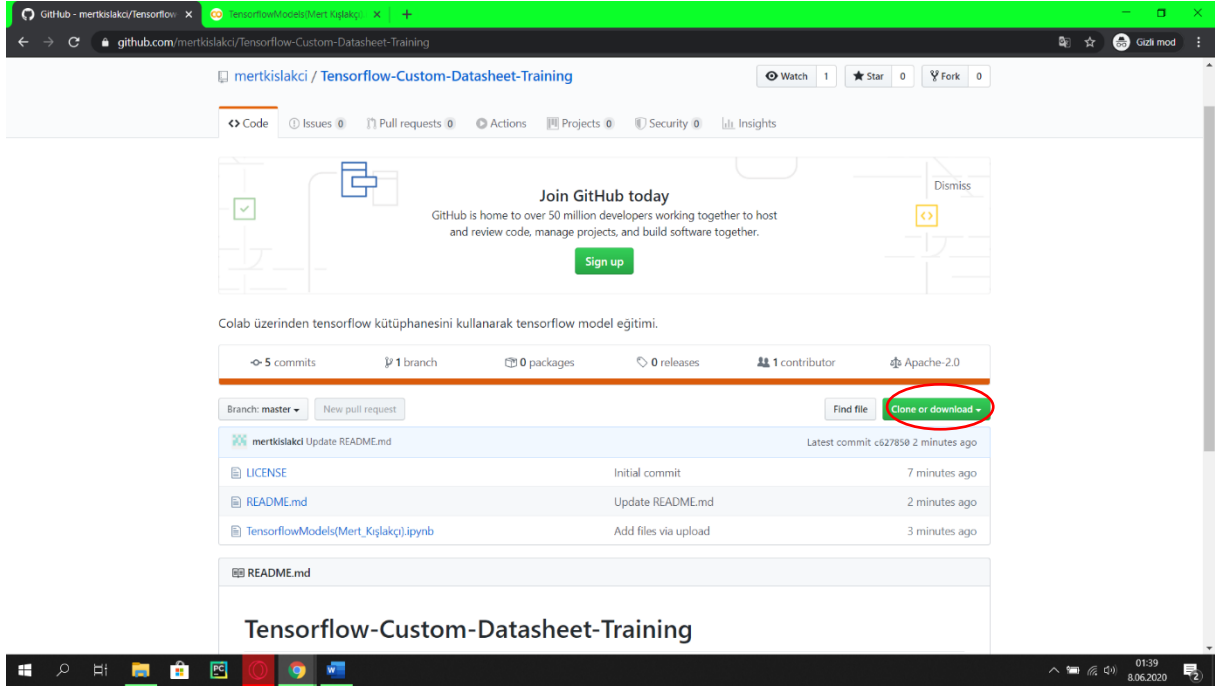
Buradan sonra artık toplanan datasheetlerin tensorflow kütüphanesi aracılığıyla eğitme işlemini gerçekleştireceğiz bu eğitimi Google Colab üzerinden gerçekleştireceğiz bu sayede hızlı bir model geliştireceğiz vereceğim github linkini Colab üzerinden açacağız.



## Modelin Colab Üzerinden Açılması ve Eğitimin Gerçekleştirilmesi

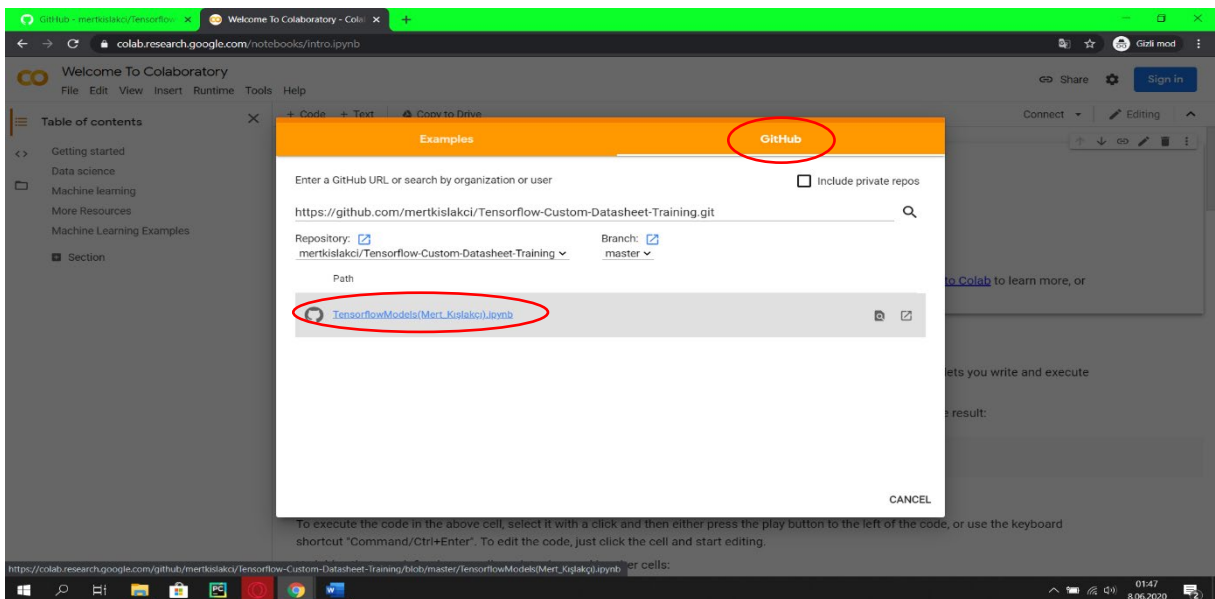
<https://github.com/mertkislakci/Tensorflow-Custom-Datasheet-Training>

Yukarıdaki link üzerinden github da bulunan Colab için gerekli olan düzenlediğim koda erişebilirsiniz biz bu link üzerinden colabı açarak kodların Colab üzerinden görülmesini sağlayacağız bunun için öncelikle “Clone or download” kısmına basarak açılan HTTPS linkini kopyalıyoruz.



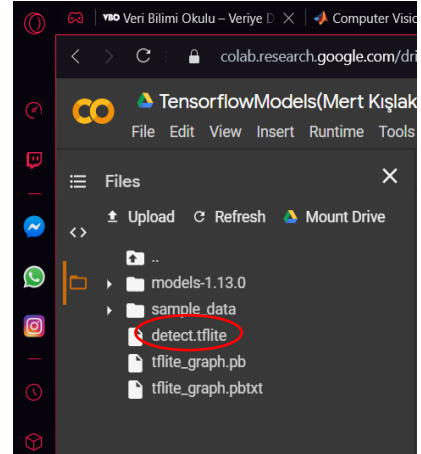
Daha sonra isterseniz Google Colab yazarak tarayıcınız üzerinden Colaba girin isterseniz de bu link üzerinden direkt Colaba erişebilirsiniz. <https://colab.research.google.com/notebooks/intro.ipynb>

Colab ekranı geldikten sonra “File”, “Open Notebook” kısmına tıklayarak aşağıda yer alan ekrandan da olduğu gibi “GitHub” sekmesine geçiyoruz ardından kopyaladığımız linki buraya yapıştırarak altta çıkan “TensorflowModels(Mert\_Kislakci).ipynb” eklentisine tıklayarak Colab ekranına geçiyoruz.



Model eğitiminin bundan sonraki adımları kod ekranında detaylıca açıklanmıştır iyi bir model eğitmek istiyorsanız topladığınız resimlerin fazlalığı çeşitliliği birbirine benzememesi gibi unsurları göz önüne almalısınız adım sayınız ne kadar fazla olursa model de o kadar iyi eğitilmiş olur. En az 5000 adımı tavsiye ederim.

Eğer modelinizi başarılı bir şekilde eğittiyseniz yandaki görselde de görüleceği üzere “detect\_tflite” uzantılı bir dosya elde edecekseniz bu dosyaya çift tıklayarak bilgisayarınıza indirin ardından “Android Studio” aracılığı ile apk şeklinde telefonumuza aktarımını yapacağız. Bu link üzerinden “Android Studio” indirebilirsiniz: <https://developer.android.com/studio>



### Modelin Android Studio Aktarımı

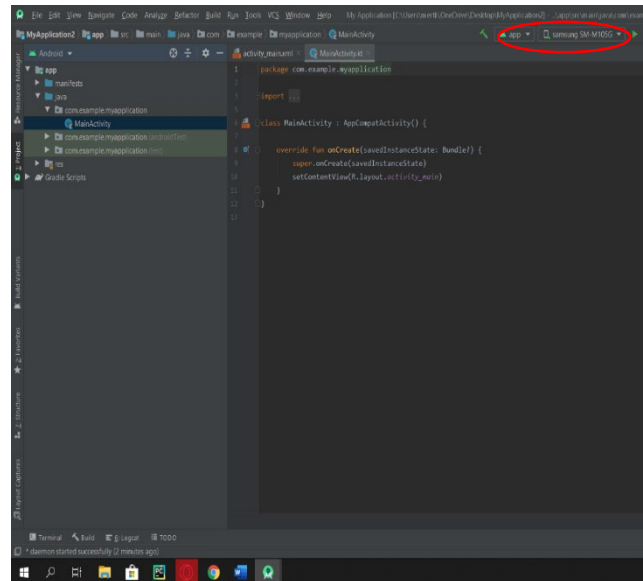
Android studio kurduğumuza göre yine tensorflowun geliştirdiği tensorflow-lite için hazır apk dosyasını ilk olarak bilgisayarımıza indiriyoruz bu dosyanın linki:

<https://github.com/tensorflow/examples>

Telefonlarımızın ayarlar kısmından geliştirici seçenekleri açıyoruz (her telefonda bu farklı olabilir) ve “USB hata ayıklaması” etkinleştirerek telefonumuzu bilgisayarımıza bağlayarak android studio açıyoruz (uyarı olarak hata ayıklamasına evet diyebilirsiniz)

New Project diyerek örnek bir android projesi açıyoruz bu sırada telefonunuza bağlanmak için izin isteyebilir onayladıktan sonra yandaki görseldeki gibi telefonun ismi çıkmaktadır, open Project diyerek

Masaüstüne çıkarttığımız tensorflow-lite dosyasını android studioda açıyoruz.

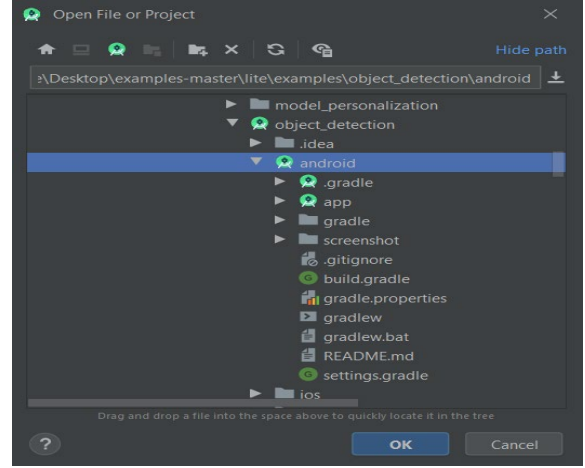


Dosyaları açtıktan sonra sırasıyla:  
examplesmaster\lite\examples

\object\_detection

\android\app\build.gradle

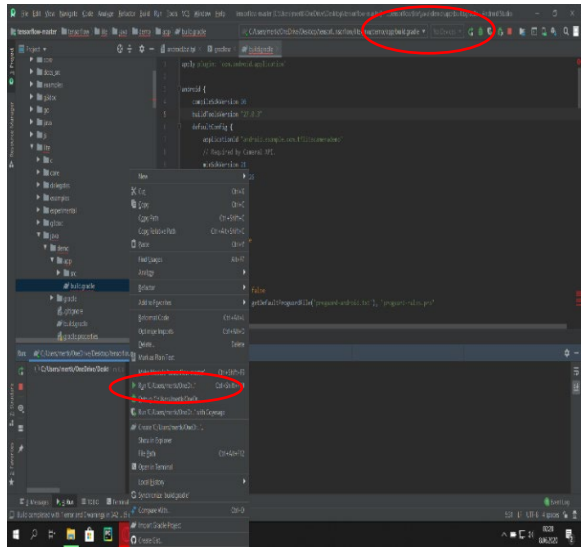
Kısmına geliyoruz burada bir süre android  
studionun dosyaları yüklemesi için bekliyoruz  
dosyaların yüklenip yüklenmediği anlamak için  
sağ alt kısımdaki daire içinde telefonunuzu  
tanıması lazım telefonu tanıdıktan sonra



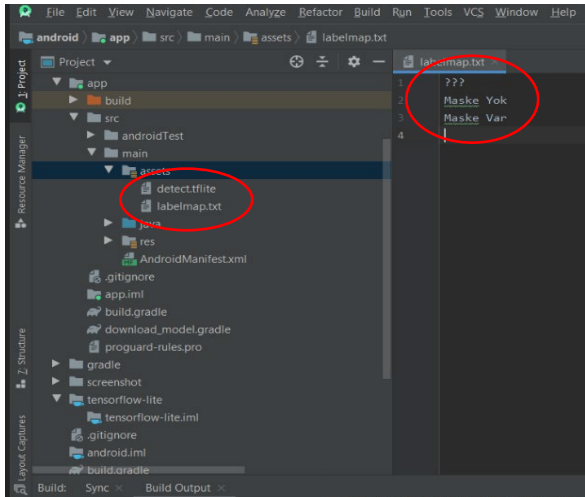
build.gradle dosyasına sağ tıklayarak runa  
basarak örnek apk telefonumuzda  
çalıştırıyoruz (bu işlemde biraz sürebilir)  
uygulama telefonumuza sıkıntısız bir şekilde  
yüklendiyse sıra geldi Colab üzerinde  
eğittiğimiz modeli yüklemeye detect.tflite  
modelini yüklemeye bunun için bu dosya  
yolunu kullanarak:

master\lite\examples\object\_detection

\android\app\src\main\assets dosya yoluna  
geliyoruz



Yan taraftaki fotoğrafta da görüleceği üzere  
assets dosyasına detect.tflite dosyasını  
siliyoruz silme işleminden eğittiğimiz model  
olan detect.tflite aynı yere yükledikten sonra  
labelmap.txt dosyasını açarak model de  
kullandığımız sınıfları sırasıyla buraya  
yazıyoruz.

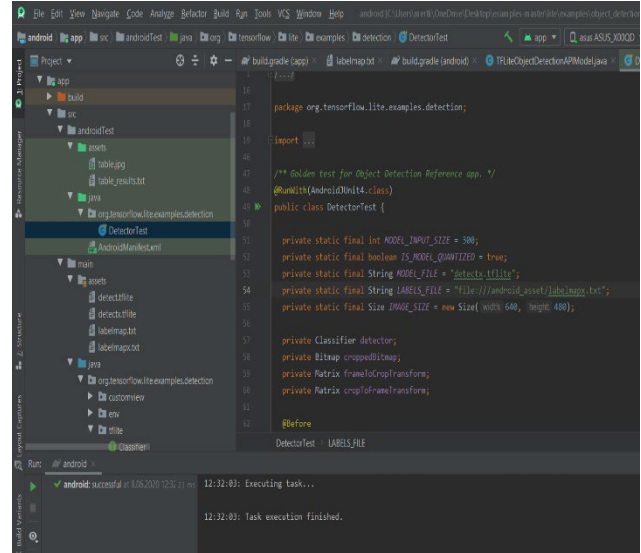




examplesmaster\lite\examples\  
object\_detection\android\app\src\  
main\java\org\tensorflow\lite  
\examples\detection\DetectorActivity.java

Telefonumuzda yer alan apk kaldırıyoruz  
kaldırma işlemi bittikten sonra tekrar  
bilgisayara bağlayarak  
DetectorActivity.javaya sağ tıklayarak run  
diyoruz ardından modelimiz yükleniyor.

Ve kendi datashettimizle eğittiğimiz  
modelimiz nesne tespiti yapabilir hale  
geliyor...



```
1 package org.tensorflow.lite.examples.detection;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.support.v4.app.ActivityCompat;
6 import android.support.v4.content.ContextCompat;
7 import android.util.Log;
8 import android.view.View;
9 import android.view.ViewGroup;
10 import android.widget.Button;
11 import android.widget.ImageView;
12 import android.widget.TextView;
13
14 import org.tensorflow.lite.Interpreter;
15 import org.tensorflow.lite.examples.detection.R;
16
17 public class DetectorTest extends Activity {
18     private static final int MODEL_INPUT_SIZE = 300;
19     private static final boolean IS_MODEL_QUANTIZED = true;
20     private static final String MODEL_FILE = "detect.tflite";
21     private static final String LABELS_FILE = "file:///android_asset/labels.txt";
22     private static final Size IMAGE_SIZE = new Size(480, 480);
23
24     private Classifier detector;
25     private Bitmap croppedImage;
26     private Matrix cropToFullTransform;
27     private Matrix cropToSmallTransform;
28
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_main);
33
34         Button btnRun = findViewById(R.id.btn_run);
35         btnRun.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick() {
38                 run();
39             }
40         });
41     }
42
43     private void run() {
44         // ... (rest of the run method code)
45     }
46 }
```

