# الجلسة العاشرة

## Deep Learning

### CNN deep network 1.1

سنقوم ببناء شبكة CNN وتدريبها على مجموعة بيانات mnist واختبارها على تصنيف الصورة إلى الصنف الصحيح الموافق.

أولاً نحتاج تنصيب هذه المكاتب

```
pip install tensorflow keras sklearn seaborn matplotlib numpy
pip install protobuf==3.20.0
```

والآن ننفذ الكود التالي:

```
import numpy as np
import matplotlib.pyplot as plt
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPooling2D, Dropout
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.models import load_model
# Define the path to the dataset
dataset_path = '.'
# Create an image data generator object
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.3)
# Load the training data
train_data = datagen.flow_from_directory(
    dataset_path,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
```

أهم مكتبة هي مكتبة Keras وهي مكتبة التعلم العميق وتتضمن العديد من التوابع الجاهزة اللازمة لبناء، وتدريب واختبار نماذج التعلم العميق

بناء Data Generator لتوليد مجموعتي التدريب والاختبار وتجهيزها للتدريب. يتم فيه تطبيع البيانات بتقسيم قيم الصور على 255 وتحديد نسبة مجموعة التحقق ب 30%

توليد train_data بمارامترات التدريب التالية: حجم الصورة 150*150، حجم الدفعة batch size هي 32 (عدد الصور المأخوذة للتدريب دفعة واحدة على GPU) Class_mode نوع التصنيف هو متعدد الفئة (3 أصناف) تحديد أن هذا الجزء يمثل مجموعة التدريب وضبط عملية قراءة الصور على وضع shuffle بحيث يتم في كل تكرار خلط بيانات التدريب

```
    subset='training',
    shuffle='true'
)
# Load the validation data
validation_data = datagen.flow_from_directory(
    dataset_path,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
# Build the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    MaxPooling2D((2, 2)),
    Dropout(0.25),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Dropout(0.25),
    Conv2D(128, (3, 3), activation='relu'),
    Flatten(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dense(3, activation='softmax')
])
```

توليد validation_data بپارامترات التدريب التالية: حجم الصورة 150*150، حجم الدفعة batch size هي 32 (عدد الصور المأخوذة للتدريب دفعة واحدة على GPU)
Class_mode نوع التصنيف هو متعدد الفئة (3 أصناف)
تحديد أن هذا الجزء يمثل مجموعة التحقق
لا نقوم بخلط بيانات التحقق

بناء نموذج شبكة CNN
طبقة convolution بحجم دخل مساوي لحجم صورة الدخل وعدد مرشحات 32 وأبعاد مرشح 3*3 وتابع تفعيل Relu
طبقة maxpooling 2*2
طبقة إلغاء dropout بنسبة إلغاء 25%
طبقة convolution بعدد مرشحات 64 بأبعاد مرشح 3*3 وتابع تفعيل Relu
طبقة Maxpooling 2*2
طبقة إلغاء Dropout بنسبة D25%
طبقة Convolution بعدد مرشحات 128 بأبعاد مرشح 33* وتابع تفعيل Relu
طبقة تسطيح لتحويل مصفوفة السمات الأخيرة لشعاع سمات
قسم التصنيف:
طبقة إلغاء Dropout 50%
طبقة كاملة الأتصال Dense بعدد عصبونات 128 وتابع تفعيل Relu
طبقة كاملة الأتصال Dense بعدد عصبونات 3 وتابع تفعيل Softmax وهي طبقة التصنيف النهائية

```
# Define the checkpoint and early stopping
checkpoint = ModelCheckpoint('best_model.h5', monitor='val_loss', mode='min', save_best_only=True)
early_stopping = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(train_data, epochs=20, validation_data=validation_data, callbacks=[checkpoint, early_stopping])
```

تحديد بارامترات تدريب النموذج:
حفظ حالة التدريب الأفضل فقط والاعتماد في عملية الحفظ على مراقبة قيمة Val_loss
إيقاف عملية التدريب في حال مرور تكرارين متتاليين بدون تحسن في قيمة Val_loss
عملية Compile وتتضمن تحديد Optimizer وهو Adam ونوع تابع loss وهو Categorical Cross Entropy ومقياس الأداء هو الدقةAccuracy
تدريب النموذج باستخدام تعليمة Fit حيث نمرر لها بيانات التدريب والتحقق و Callbacks

```
# load the best saved model
model = load_model('best_model.h5')
# Evaluate the model
loss, accuracy = model.evaluate(validation_data)
print(f'Test accuracy: {accuracy}')
```

بعد انتهاء عملية التدريب يتم:
تحميل النموذج الذي تم حفظه وهو يمثل أفضل حالة تدريب
تقييم أداء النموذج المدرب من خلال اختبار مجموعة التحقق والحصول من نتيجة الاختبار على الدقة وال loss وطباعتها

```
# Generate predictions
y_pred = []
y_true = []
for i in range(len(validation_data)):
  X, y = validation_data[i]
  y_pred.extend(np.argmax(model.predict(X), axis=-1))
  y_true.extend(np.argmax(y, axis=-1))
# Print our model's predictions
print("These are the model predictions :")
print(y_pred)
# Check our predictions against the ground truths
```

أخذ كل عينة من عينات التحقق واختبار الموديل المدرب بها ثم مقارنة الأصناف التي تم التعرف عليها predictions مع الأصناف الحقيقية لكل عينات التحقق لتحديد مقاييس الأداء

```
print("These are the corresponding labels :")
print(y_true)
# Plotting the confusion matrix
confusion_mtx = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10,8))
sns.heatmap(confusion_mtx, annot=True, fmt="d")
plt.title("Confusion matrix")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
# Classification report
print("Classification Report: \n", classification_report(y_true, y_pred))
# Plot the training and validation accuracy
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

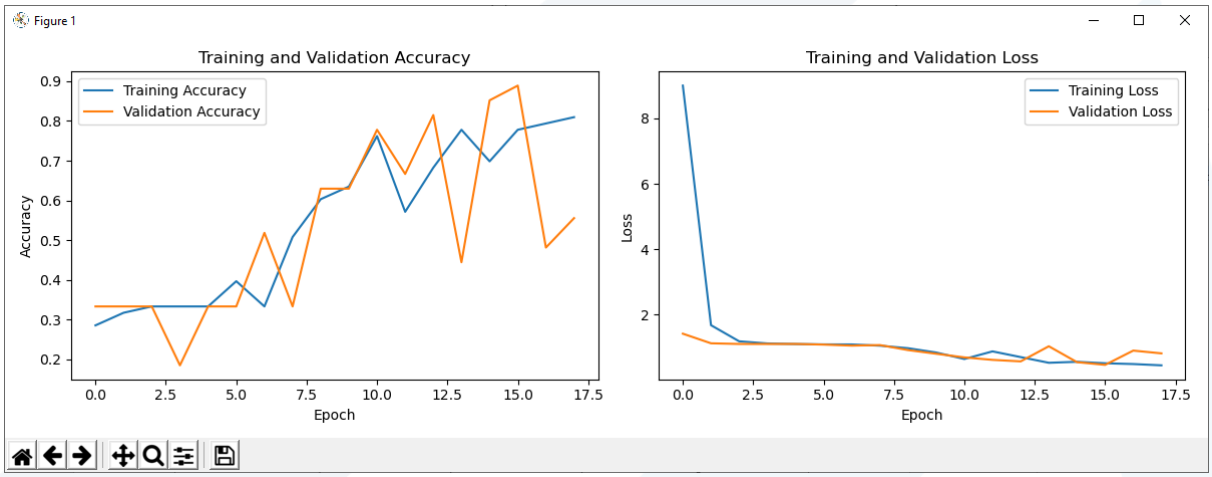
طباعة مصفوفة التشتت Confusion Matrix
وتقرير التصنيف ويتضمن قيم precision, recall, f1-score, accuracy للأصناف الثلاثة
إضافة للقيم المتوسطة لكل معامل منها

```
# Plot the training and validation loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
```

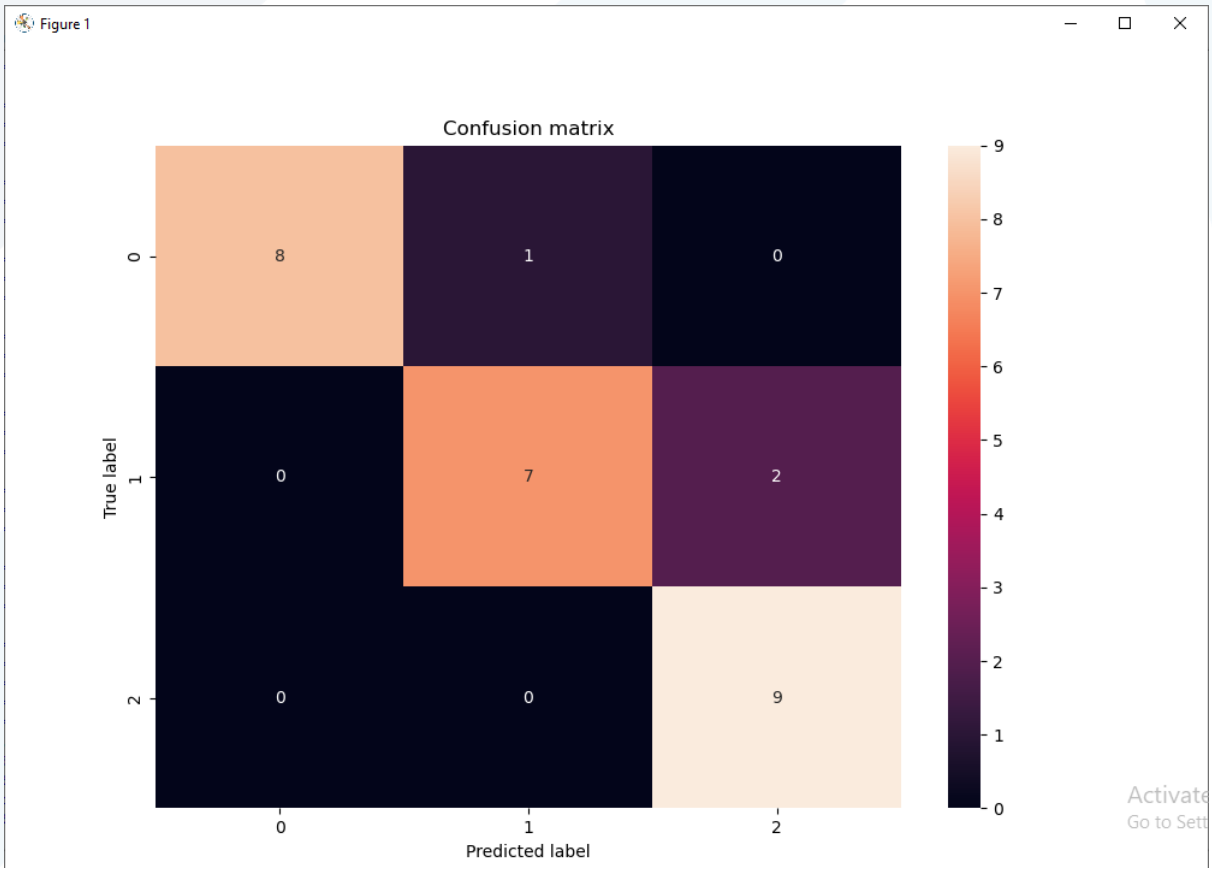
رسم منحنيات الدقة و Loss لمجموعتي التدريب والتحقق

```
plt.legend()

plt.tight_layout()

plt.show()
```

منحنيات الدقة والloss لعملية التدريب:




مصفوفة التشتت الناتجة عن اختبار الموديل المدرب على مجموعة بيانات الاختبار



دقة الاختبار وقيم Precision, recall, f1-score للأصناف الثلاثة:

```
Test accuracy: 0.8888888955116272
These are the model predictions :
[0, 2, 2, 1, 1, 2, 0, 1, 2, 1, 2, 1, 2, 0, 1, 2, 2, 0, 1, 0, 1, 0, 0, 2, 2, 0, 2]
These are the corresponding labels :
[0, 1, 1, 1, 1, 2, 0, 1, 2, 1, 2, 1, 2, 0, 0, 2, 2, 0, 1, 0, 1, 0, 0, 2, 2, 0, 2]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.89      0.94         9
           1       0.88      0.78      0.82         9
           2       0.82      1.00      0.90         9

    accuracy                           0.89        27
   macro avg       0.90      0.89      0.89        27
weighted avg       0.90      0.89      0.89        27
```

كود اختبار الموديل على صور اختبار مستقلة:

بعد تدريب الموديل وحفظه في المسار الحالي يمكن استخدام الموديل المدرب وتحميله واختباره بعينات اختبار دون الحاجة لإعادة التدريب مرة أخرى.

نفذ الكود:

```
import matplotlib.pyplot as plt
from keras.preprocessing.image import load_img, img_to_array
from keras.models import load_model
import numpy as np
# Load the image
img = load_img('test1.jpg', target_size=(150, 150))
# Convert the image to a numpy array
img_array = img_to_array(img)
# Expand dimensions so the image has the same shape as the training set
img_array = np.expand_dims(img_array, axis=0)
# Normalize the image
img_array /= 255.
# Load the saved model
model = load_model('best_model.h5')
# Make a prediction
prediction = model.predict(img_array)
```

تنصيب المكاتب اللازمة والأهم Keras

تحميل صورة الاختباروتحويلها لمصفوفة

إضافة بعد إضافي لصورة الاختبار لتلائم شكل الدخل الخاص بالموديل المدرب

بتقسيم قيم الصورة على 255normalization ثم تطبيق عملية

اختبار الموديل باستخدام تابع predict للنموذج المدرب ثم أخذ القيمة العظمى من ناتج predict كون أنَّ تابع طبقة التصنيف هو softmax ويعطي احتماليات للأصناف الثلاثة لذلك نطبق عملية argmax لحساب الاحتمالية الأعلى التي تمثل الصنف الذي تم التعرف عليه

```
# Get the class with the highest probability
predicted_class = np.argmax(prediction)
# Get the prediction probability
predicted_probability = np.max(prediction)
# Define a dictionary to map the indices to the soil types
class_dict = {0: 'Cinder', 1: 'Cracked', 2: 'Sandy'}
# Get the soil type
predicted_soil_type = class_dict[predicted_class]
# Display the image with the predicted class and probability as the title
plt.imshow(img)
plt.title(f'{predicted_soil_type} (Probability: {predicted_probability:.2f})')
plt.show()
```

عرض صورة الاختبار مع الصنف الذي تم التعرف عليه مع درجة المطابقة (احتمالية التصنيف) وهي تمثل نسبة احتمالية أن تكون العينة هي من الصنف المذكور

والنتيجة:

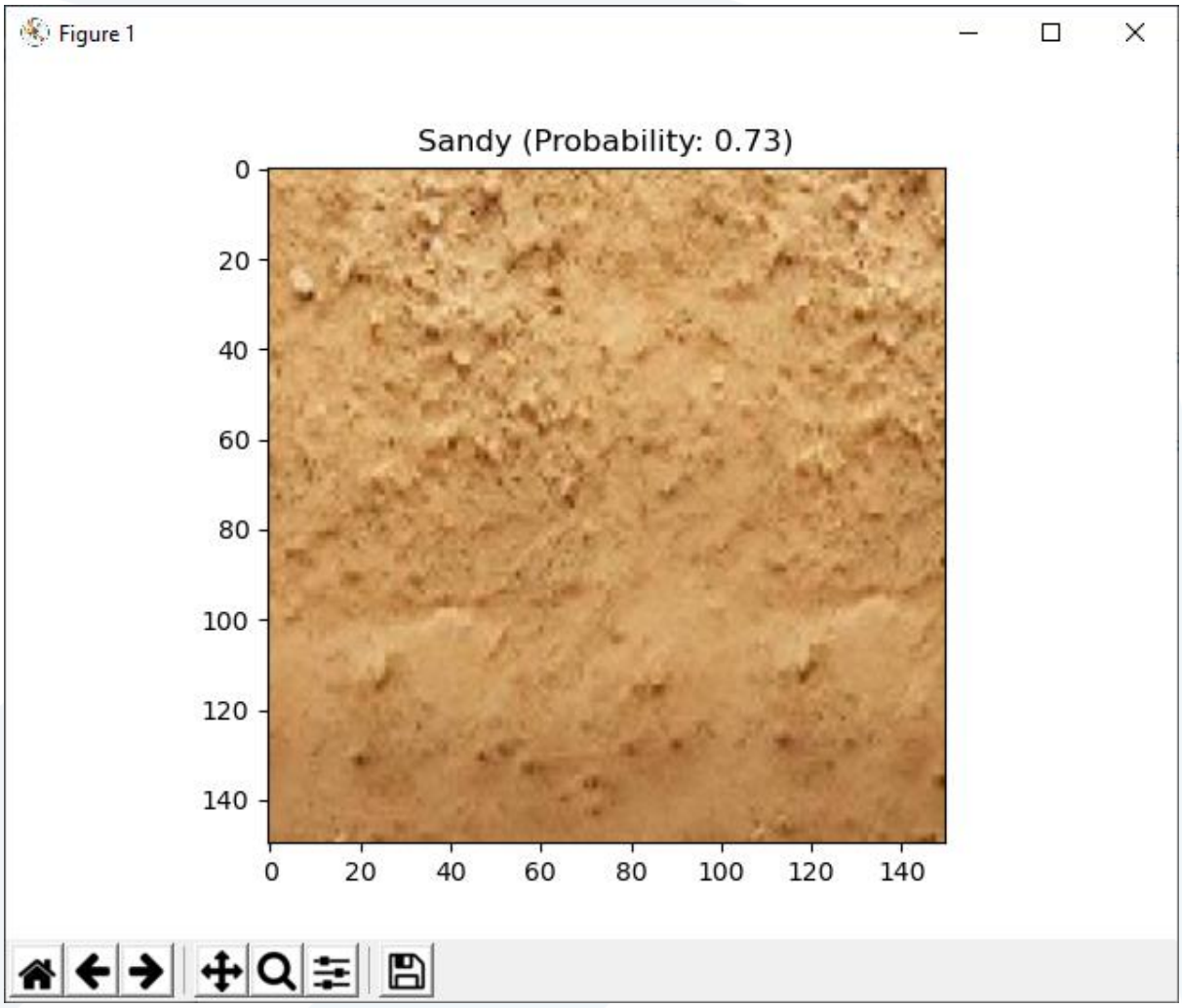


Figure 1
Sandy (Probability: 0.73)
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

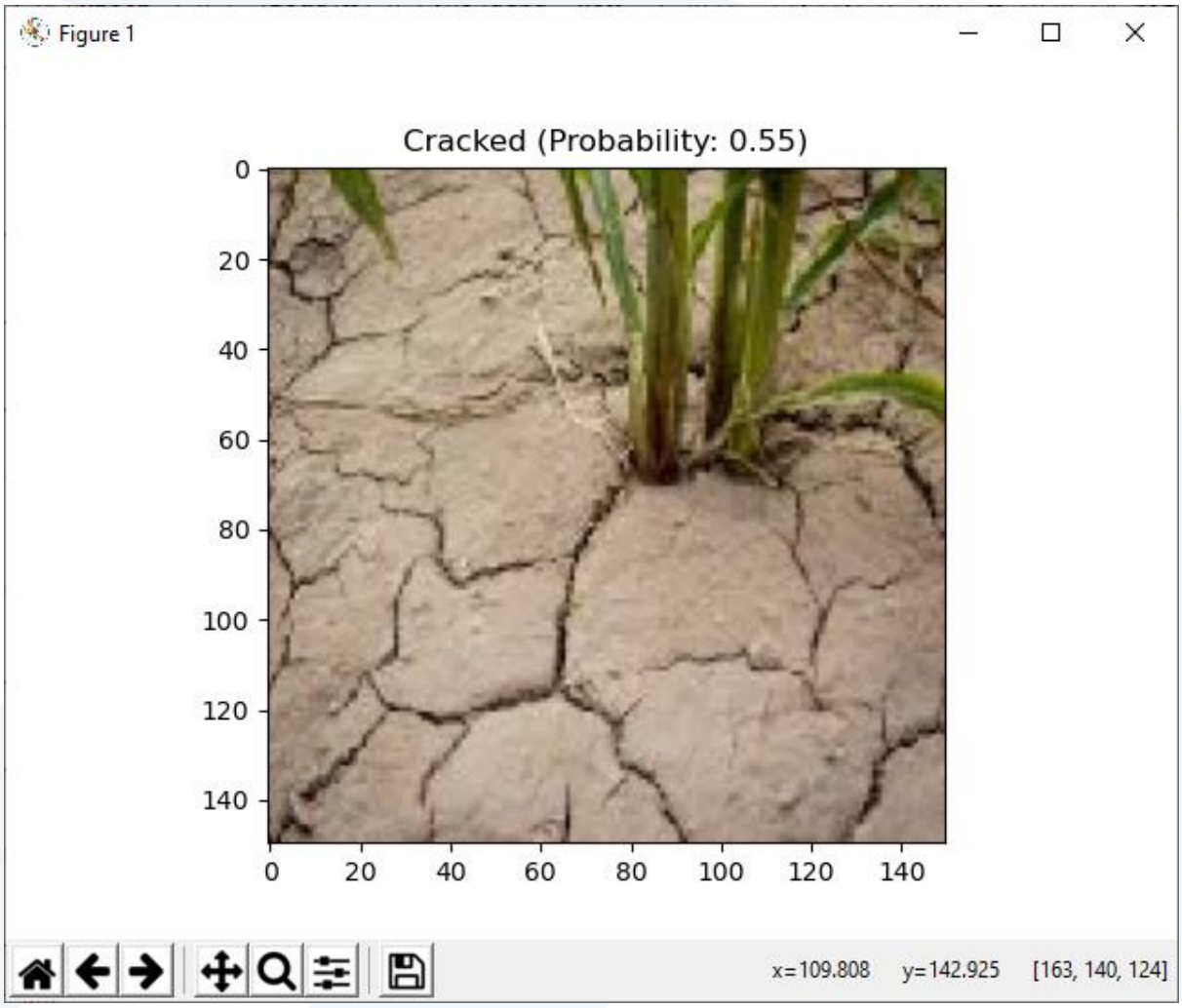


Figure 1
Cracked (Probability: 0.55)
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140
x=109.808
y=142.925
[163, 140, 124]

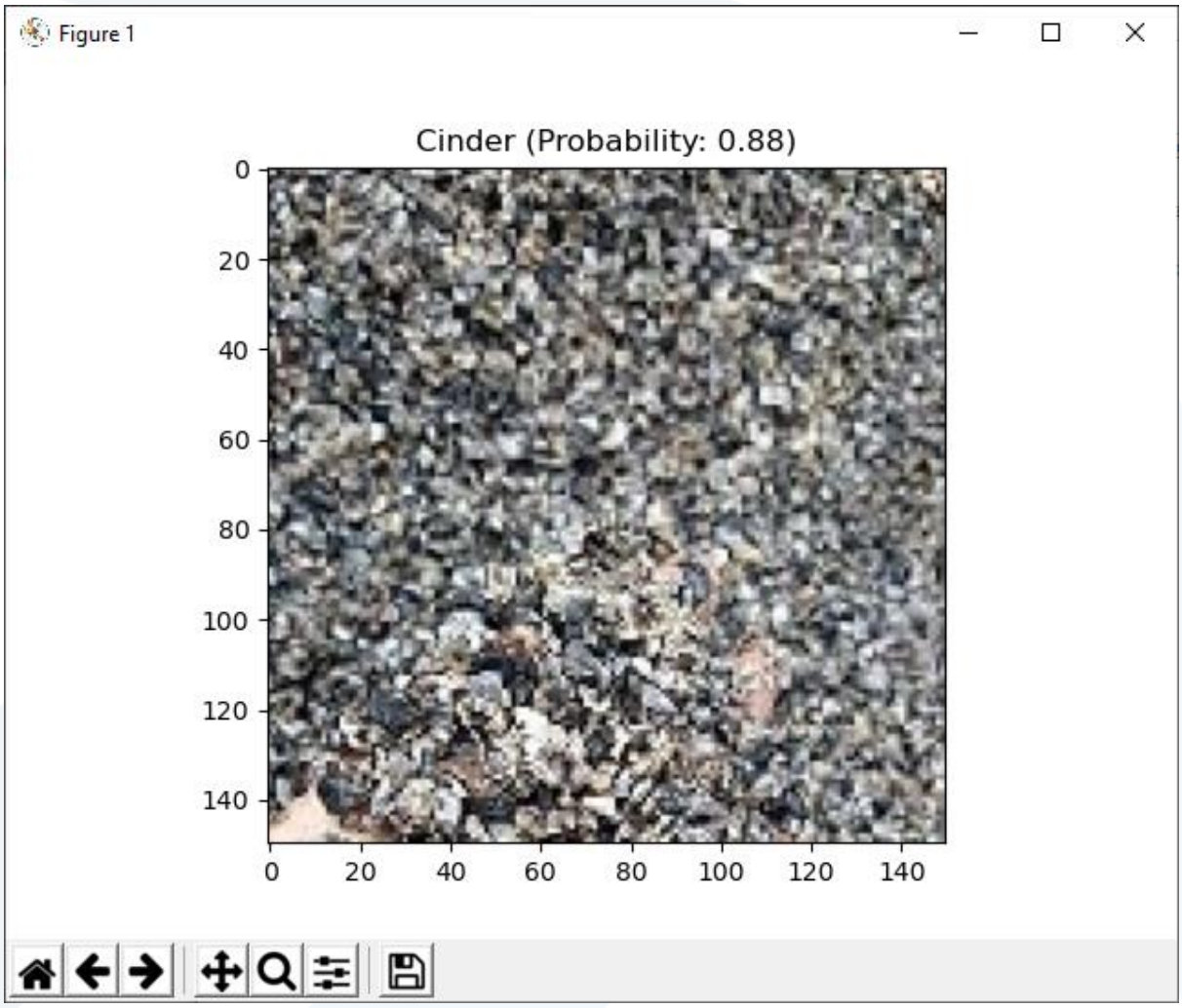


Figure 1
Cinder (Probability: 0.88)
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140