الباحث: نوار بشار حسن، سوار علي مصطفى، زينب أنيس عطاف

اكتشاف أمراض الجلد (وحمة، سرطان الجلد، التقرن الدهني) باستخدام التعلم العميق

ملخص

يعد التعلم العميق أحد العلوم المتقدمة التي تعتمد على الشبكات العصبية الصناعية، والذي شكل قفزة نوعية في تطوير حياة البشر وطريقة تفاعلهم مع الآلة، إذ تمكنت خوارزميات التعلم العميق من بلوغ دقة عالية في مختلف تطبيقات التصنيف والعنقدة، إلا إنه يحتاج إلى قواعد بيانات ذات حجم كبير ووقت تدريب أطول وأجهزة حاسوب ذات إمكانيات عالية. ويستخدم في العديد من المجالات العملية ومن أهمها التعرف على الامراض مثل الامراض الجلدية، وخاصة أن الامراض الجلدية تحتاج الى دقة كبيرة من اجل التعرف الصحيح عليها، لذلك أصبح التعلم العميق أداة مساعدة في المجالات الطبية بشكل عام والتعرف على الامراض وفرزها حسب نوع كل حالة بشكل خاص.

أمراض الجلد المدروسة هي الوحمة Nevus وسرطان الجلد Melanoma والتقرن الدهني Seborrheic Keratosis. لكل نوع من الامراض الجلدية مجموعه من الخصائص والاعراض لذلك يمكن ان يستخدم التعلم العميق من اجل تمييز الامراض الجلدية عن طريق شكل هذه الامراض.

الكلمات المفتاحية: التعلم العميق، الامراض الجلدية.

Detecting skin diseases (nevus, melanoma, seborrheic keratosis) using deep learning

Abstract

Deep learning is one of the advanced sciences that relies on artificial neural networks, which constituted a quantum leap in the development of human life and the way they interact with the machine, as deep learning algorithms were able to achieve high accuracy in various classification and clustering applications, but it needs large-sized databases, longer training time and high-powered computers. It is used in many practical fields, the most important of which is the identification of diseases such as dermatology, especially that skin diseases require great accuracy in order to correctly identify them, so deep learning has become an aid in medical fields in general and to identify diseases and sort them according to the type of each case in particular .

The studied skin diseases are nevus, melanoma and seborrheic keratosis. Each type of skin disease has its own set of characteristics and symptoms, so deep learning can be used to distinguish skin diseases by their form.

Keywords: deep learning, skin diseases.

مقدمة

تعد خوارزميات التعلم العميق ثورةً في مجال الذكاء الصناعي، لأنها قللت عبء استخلاص السمات من قبل الباحثين. إذ تقوم خوارزميات التعلم العميق باستخراج السمات بشكل تلقائي، مما يؤدي إلى زيادة دقة أنظمة التعرف المعتمدة على التعلّم العميق. إن استخدام البرمجيات الالكترونية الحديثة في التعرف على الامراض وفرزها يعد ضرورة مهمه جداً بسبب الدقة الكبيرة التي تقدمها هذه التقنيات الحديثة ومعدل الخطأ المنخفض.

الأمراض الجلدية هي الأمراض التي تصيب جلد الإنسان وقد تكون هذه الأمراض معدية أو غير معدية بحسب نوع المرض، حيث إن الجلد هو من أكثر الأعضاء في الجسم عرضةً للتأثيرات الخارجية والبيئية، أمراض الجلد هي تحولات في الجلد مثل الوحمة Nevus وسرطان الجلد Melanoma والتقرن الدهني Seborrheic Keratosis.

يمكن استخدم تقنيات الذكاء الاصطناعي مثل التعلم الآلي والتعلم العميق من اجل اكتشاف أمراض الجلد.

أهمية البحث وأهدافه:

يعمل هذا البحث على تطبيق تقنيات التعلم العميق من اجل فرز الصور الامراض الجلدية الى ثلاث أصناف (nevus, melanoma, seborrheic keratosis). بالإضافة الى استخدام واحدة من اكثر شبكات التعلم العميق شهرةً (VGG16) من خلال اجراء تعديل بسيط في بنية الشبكة من أجل ان تلائم هدف البحث.

أدوات البحث:

- لغة البرمجة البايثون
- مكتبات Kears & Tensorflow
- موقع Colab
- شبكة VGG16

1- مفهوم التعلم العميق Deep learning

هناك عدة تعريفات للتعلم العميق:

1- صف من تقنيات تعلم الآلة التي تستخدم عدة طبقات من المعالجة غير الخطية للمعلومات لاستخراج ونقل الخواص بإشراف أو دون إشراف، بالإضافة إلى التحليل والتصنيف للأنماط.

2- حقل فرعي ضمن تَعلّم الآلة يُطبق خوارزميات التعلم على تمثيل متعدد المستويات، وذلك من أجل نمذجة العلاقات المعقدة ضمن المعطيات، وبذلك يتم تعريف السّمات والمفاهيم العالية المستوى بناءً على ما هو أدنى منها (فالسيارة مثلا تعرّف بأجزائها، والدولاب يُعرّف بأجزائه..... وهكذا)؛ وتسمى هذه البنية بالبنية العميقة.

3- مجموعة من خوارزميات تعلّم الآلة التي تحاول أن تتعلم في عدّة مستويات مقابلة لمثلها من التجريد، ويستعمل التعلّم العميق الشبكات العصبونية الصنعية. وتتوافق المستويات في هذه النماذج التعلّمية الإحصائية مع مستويات مختلفة من المفاهيم إذ يتم تعاريف المفاهيم عالية المستوى بالاعتماد على المفاهيم ذات المستوى الأدنى، وقد تساعد مفاهيم المستوى الأدنى ذاتها في تعريف العديد من مفاهيم المستوى الأعلى.

ونلاحظ من التعاريف السابقة وجود مفهومين أساسيين مشتركين، وهما:

1- وجود نماذج مؤلفة من عدة مستويات للمعالجة غير الخطية للمعلومات.

2- طرائق تعلّم تمثيل للسمات – بإشراف أو دون إشراف – أكثر تجريداً في المستويات الأعلى.

يقع التعلّم العميق في المنطقة المشتركة ما بين عدّة مجالات وهي بحوث الشبكات العصبونية والذكاء الصنعي والنمذجة الرسومية والأمثلية والتعرّف على الأنماط ومعالجة الإشارات. وهناك ثلاثة أسباب وراء انتشار التعلّم العميق وشيوعه، هي:

1- القدرات المتزايدة في معالجة الرقاقات كوحدات المعالجة الرسومية عامة الأغراض general purpose graphical processing units (GPGPUs).

2- الحجم المتزايد للمعطيات المستخدمة في التدريب.

3- التقدّم الحديث في تعلّم الآلة ومعالجة الإشارة والمعلومات.

لقد سمحت هذه التطورات الأخيرة لوسائل التعلّم العميق باستعمال التوابع المعقّدة المركبة غير الخطية من أجل تعلّم تمثيلات الخواص الموزّعة والهرمية، وبتحقيق الاستعمال الفعّال للمعطيات المصنّفة وغير المصنّفة.

كما أثبتت دراسات العديد من الخبراء في عدّة جامعات نجاح التعلّم العميق في تطبيقات متنوّعة مثل الرؤية الحاسوبية، والتعرّف اللفظي، والبحث الصوتي والتعرّف إلى الخطاب، وترميز الصفات الصوري والخطابي، والتصنيف الدلالي للكلام، وفهم اللغات الطبيعية، والتعرّف إلى خط اليد.

1-1 أنواع شبكات التعلّم العميق

1- الشبكات العميقة للتعلّم الذاتي أو دون إشراف:

تستخدم عند عدم توافر معلومات عن عدد صفوف الهدف أو تسمياتها؛ إذ تقوم هذه الشبكات بالتقاط الترابط العالي المستوى لمعطيات الدخل محاولةً تحليله والكشف عن أنماط في هذه المعطيات.

2- الشبكات العميقة للتعلّم مع إشراف:

هي ممتازة في تصنيف الأنماط، وذلك عبر توصيف الصفوف الهدف بدقة وفقاً للمعطيات المرئية.

وتكون البيانات الهدف المعرّفة متوفرة دائماً سواء بشكل مباشر أو غير مباشر.

3- الشبكات العميقة الهجينة:

يكون هدفها تميزي ويطبق على ناتج الشبكات العميقة المولّدة أو غير المراقبة. وتكون الشبكات في هذا المجال النوع الأكثر جودة من شبكات النوع السابق.

1-2 الفرق بين التعلم العميق وتعليم الآلة

يُعدّ التعليم العميق (DL) وتعلّم الآلة Machine Learning (ML) فرعان من فروع الذكاء الاصطناعي لكن بينما يسمح تعلم الآلة للحواسيب أن تتعلم بنفسها من خلال التقاط وتحليل كميات ضخمة جداً من البيانات وتدريب النظام عليها ثم اتخاذ القرار، فإن التعلّم العميق يركز على فهم الآلة أو الحاسب للأمور بمستوى عالٍ من الفهم والإدراك لذا فهو يقوم بتحليل كميات أكثر ضخامة من تعلّم الآلة وأكثر تعقيداً وذلك بغية الغوص أكثر في عملية اتخاذ القرار. مثلاً إذا كنا نريد تطوير تطبيق يفهم مشاعر الناس التي تظهر على وجوههم من خلال صور الوجوه عندها تكون مهمة ML هي إدخال آلاف صور الوجوه الى النظام في حين يكون تركيز DL على تمييز الأنماط

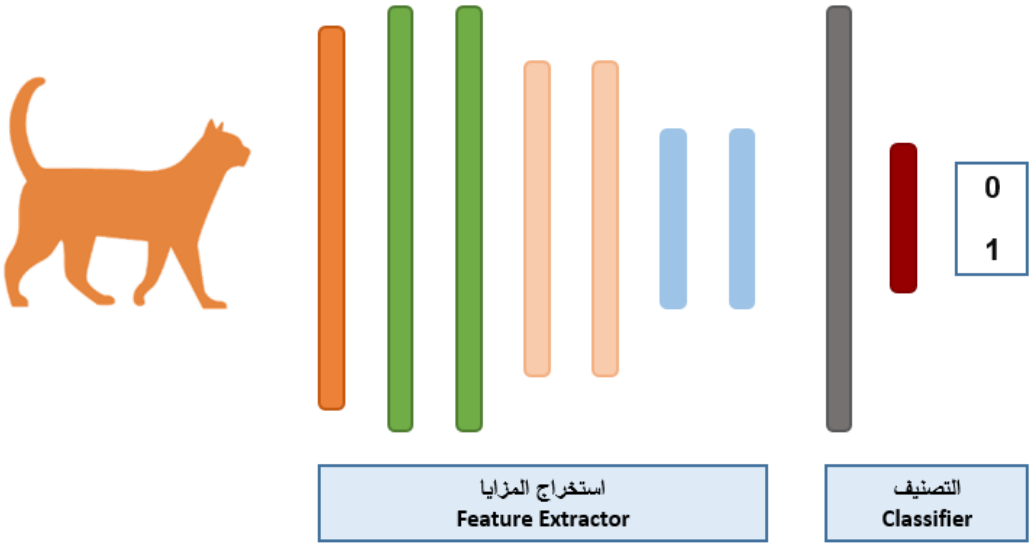
في الوجوه لكشف المشاعر من خلالها. لذا يمكن القول إن مفهوم DL أعمق وأكثر تحديداً من مفهوم ML أو يمكن القول إن ML مفهوم أعم أما DL فهو مفهوم أكثر تحديداً وخصوصيةً

2- نقل التعلم

نقل التعلم هو تقنية يتم تطويرها في مجال تعلم الآلة وتعنى بحفظ المعلومات التي يتم اكتسابها في محاولة حل مسألة معينة لغرض استخدامها في حل مسألة أخرى مشابهة لها. فعلى سبيل المثال المعلومات المكتسبة في التعرف على صور الطيور يمكن استخدامها عند محاولة التعرف على صور الطائرات. وكمثال آخر يمكن استخدام المعلومات المكتسبة في التعرف على السيارات في التعرف على أشياء أخرى مشابهه لها مثل الشاحنات.

2-1 كيف يستفيد نقل التعلم من الشبكات المدربة مسبقاً.

قامت العديد من المجموعات البرمجية ببناء عدة نماذج للشبكات العصبية هدفها التعرف على الصور. تتألف هذه النماذج من عدد كبير من الطبقات المخفية المعقدة في بنيتها. وقد تم تدريب هذه النماذج على أعداد هائلة جدً من الصور تقدر بالملايين. أظهرت هذه النماذج قدرة كبيرة جداً في التعرف على عدد كبير من الكائنات. يمكن إعادة استخدام هذه النماذج الذكية في التعرف على كائنات جديدة لم يتم التدريب عليها مسبقاً. وذلك من خلال استئصال الطبقة الأخيرة واستبدالها بطبقات جديدة وإعادة تدريبها لغرض تهيئتها للتعرف على كائنات جديدة. وتسمى هذه العملية "نقل التعلم".



الشكل (1)

2-2 أشهر النماذج المتاحة في نقل التعلم:

- VGG16 : ظهرت في عام 2014 بواسطة "سيمونيان و زيسرمان". تحتوي على 16 طبقة وتبلغ مساحة تخزينها 533 ميغابايت. علماً أنه في 2014 كان النموذج المكون من 16 طبقة يعتبر عميق جداً.
- ResNet50 : ظهرت في عام 2015 بواسطة "He et al.". وتتألف من 50 إلى 200 طبقة. وعلى الرغم من زيادة عدد الطبقات فيها مقارنة ب "VGG16" إلا إن مساحتها تبلغ 102 ميغابايت وذلك نظراً لاختلاف بنيتها التركيبية.
- Inception V3 : كان أول ظهور لها بتركيبتها المصغرة في عام 2014 بواسطة "سيغادي وآخرون" ، ومساحتها أقل سابقاتها حيث بلغت 96 ميغابايت فقط.

وهنا لا بد أن نعرج على المصطلح المهم والشائع في هذا السياق "ImageNet" ، وهو عبارة كمية كبيرة من البيانات تقدر بـ 1.2 مليون صورة تستخدم لتدريب هذه النماذج لتصنيف 1000 كائن. ويقام سنوياً تحدي لتقييم هذه النماذج من خلال قياس اداءها ومقارنة نتائج كل نموذج بناءً من خلال هذه البيانات. يطلق على هذه الفعالية:

"ImageNet Large Scale Visual Recognition Challenge, or ILSVRC "

3- شبكات VGG16

الشبكة العصبية VGG اسمها الكامل مجموعة الهندسة المرئية (Visual Geometry Group) وتستخدم للتعرف على الصور ، تتكون هذه الشبكة من 16 طبقة :

1. طبقات الالتفاف (3*3 Convolutions layers)
2. طبقات التجميع القصوى (2 * 2 Max pooling layers)
3. طبقات متصلة بالكامل في النهاية (Fully connected)

4- القسم العملي

أولا يجب تحديد المسار الذي توجد فيه قاعدة البيانات

```
BASE_DATASET_FOLDER = os.path.join("/content/drive/MyDrive/dataset")
```

ثم نقوم بتعريف ثلاث متغيرات تشير إلى أسماء الملفات التي تتضمنها قاعدة البيانات

```
TRAIN_FOLDER = "train"
VALIDATION_FOLDER = "validation"
TEST_FOLDER = "test"
```

أبعاد بيانات الدخل (الصور) لشبكة VGG16 هو 224 للعرض و 224 للطول، أيضا يجب الإشارة إلى أن كل صورة يجب أن تكون ناتج دمج ثلاث مصفوفات للألوان RGB

```
IMAGE_SIZE = (224, 224)
INPUT_SHAPE = (224, 224, 3)
```

يجب ضبط اعدادات الـ keras من أجل تحديد حجم الـ Batch الخاص بعملية التدريب Training والتحقق Validation

```
TRAIN_BATCH_SIZE = 64
VAL_BATCH_SIZE = 8
```

تحديد عدد الأعظمي لتكرارات عملية التدريب التي سوف يقوم فيها الـ model

```
EPOCHS = 50
```

تحديد قيمة معامل التعلم

```
LEARNING_RATE = 0.0001
```

يجب تحديد اسم الملف الذي يجب تخزين الأوزان فيه كما في الشكل التالي

```
MODEL_PATH = os.path.join("derma_diseases_detection.h5")
```

من اجل تهيئة الصور التي سوف يتم تدريب النموذج عليها يجب استخدام الكتلة البرمجية التالية التي تعمل على ضبط ابعاد الصور

```
train_datagen = ImageDataGenerator(
        rescale=1./255,
        width_shift_range=0.2,
        height_shift_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')
```

الكتلة البرمجية التالية تستخدم من اجل توسيع قاعدة البيانات (صور قاعدة البيانات المخصصة لعملية التدريب (Training

```
train_generator = train_datagen.flow_from_directory(
        os.path.join(BASE_DATASET_FOLDER, TRAIN_FOLDER),
        target_size=IMAGE_SIZE,
        batch_size=TRAIN_BATCH_SIZE,
        class_mode='categorical',
        shuffle=True)
```

الكتلة البرمجية التالية تستخدم من اجل توسيع قاعدة البيانات (صور قاعدة البيانات المخصصة لعملية التحقق (Validation

```
val_datagen = ImageDataGenerator(rescale=1./255)
val_generator = val_datagen.flow_from_directory(
```

```
        os.path.join(BASE_DATASET_FOLDER, VALIDATION_FOLDER),
        target_size=IMAGE_SIZE,
        class_mode='categorical',
        shuffle=False)
```

الكتلة البرمجية التالية تستخدم من اجل توسيع قاعدة البيانات (صور قاعدة البيانات المخصصة لعملية الاختبار Testing)

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
        os.path.join(BASE_DATASET_FOLDER, TEST_FOLDER),
        target_size=IMAGE_SIZE,
        batch_size=VAL_BATCH_SIZE,
        class_mode='categorical',
        shuffle=False)
```

الكتلة البرمجية التالية تستخدم من اجل الحصول على أسماء الاصناف التي تتكون منها قاعدة البيانات، وترميز كل صنف الى رقم معين (0,1, 2)

```
classes = {v: k for k, v in train_generator.class_indices.items()}
print(classes)
```

التعليمة التالية تستخدم من اجل توليد model من النوع VGG16 وتحميل اوزان الشبكة التي سبق تدريبها ('imagenet') وضبط شكل الصور التي تعتبر دخل للشبكة بالأبعاد 224, 224, 3

```
vgg_model = vgg16.VGG16(weights='imagenet', include_top=False, input_shape=IN
PUT_SHAPE)
```

التعليمة التالية تستخدم من اجل الاحتفاظ بأوزان كامل طبقات شبكة VGG16 ما عدا الطبقات الأربعة الأخيرة ليتم تدريب هذه الطبقات على الصور الخاصة بنا

```
for layer in vgg_model.layers[:-4]:
    layer.trainable = False
```

التعليمات التالية تستخدم من اجل بناء الموديل الخاص بنا عن طريق وضع شبكة VGG16 ثم تليها بعض الطبقات الإضافية التي قمنا بوضعها بعد VGG16 وقمنا بضبط عدد مخارج model المستخدم من اجل يكون عدد المخارج يساوي 3 بحسب عدد الأصناف المستخدمة

```
# Create the model
model = Sequential()

# Add the vgg convolutional base model
model.add(vgg_model)

# Add new layers
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(classes), activation='softmax'))
```

التعليمات التالية تستخدم من اجل ضبط بعض بارامترات عملية التدريب لل Model حيث ان تابع الخسارة هو 'categorical_crossentropy' وقيمة معامل الخطأ هي 0.0001

```
# Compile the model
```

```
from tensorflow.keras import models,optimizers,layers
model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.Adam(lr=LEARNING_RATE),
              metrics=['acc'])
```

التعليمة التالية تستخدم من اجل بدء عملية التدريب لل Model وتحديد عدد خطوات عملية التدريب وعدد خطوات عملية التحقق.

```
history = model.fit_generator(
      train_generator,
      steps_per_epoch=train_generator.samples//train_generator.batch_size,
      epochs=EPOCHS,
      validation_data=val_generator,
      validation_steps=val_generator.samples//val_generator.batch_size)
```

التعليمة التالية تستخدم من اجل حفظ الاوزان الجديدة التي تم الحصول عليها بعد الانتهاء من عملية التدريب السابقة

```
model.save("/content/drive/MyDrive/derma_diseases_detection.h5")
```

الكتلة البرمجية التالية تستخدم من اجل رسم منحنيات عملية التدريب والتحقق لكل من الدقة والخسارة (دقة الـ model في التعرف على الصور ومقدار خطأ الـ model في التعرف على الصور)

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
```

```
val_loss = history.history['val_loss']


epochs = range(len(acc))


plt.plot(epochs, acc, 'b', label='Training acc')

plt.plot(epochs, val_acc, 'r', label='Validation acc')

plt.title('Training and validation accuracy')

plt.legend()


plt.figure()


plt.plot(epochs, loss, 'b', label='Training loss')

plt.plot(epochs, val_loss, 'r', label='Validation loss')

plt.title('Training and validation loss')

plt.legend()


plt.show()
```

التعليمة التالية تستخدم من اجل الحصول على الدقة الفعلية والخسارة (الخطأ) الفعلي للنموذج السابق الذي تم تدريبه من خلال اختبار النموذج على صور الموجودة في قسم الاختبار.

```
loss, accuracy = model.evaluate_generator(test_generator,steps=test_generator.sam

ples//test_generator.batch_size)
```

التعليمة التالية تستخدم من اجل طباعة قيم الدقة والخسارة التي تم الحصول عليها في التعليمة السابقة

```
print("Accuracy: %f\nLoss: %f" % (accuracy,loss))
```

التعليمات التالية تستخدم من اجل الحصول على القيم الفعلية التي يعطيها النموذج السابق الذي تم تدريبه من خلال تخزين مصفوفة القيم الرقمية التي يعطيها النموذج لكل صورة في Y_pred ثم يتم الحصول على اكبر قيمة في كل مصفوفة ليتم وضعها في y_pred

```
Y_pred = model.predict_generator(test_generator,verbose=1, steps=test_generator.s
amples//test_generator.batch_size)
y_pred = np.argmax(Y_pred, axis=1)
```

التعليمة التالية تستخدم من اجل الحصول على مصفوفة الارتباك (confusion matrix)

```
cnf_matrix = confusion_matrix(test_generator.classes, y_pred)
```

التابع التالي يستخدم من اجل رسم شكل توضيحي يبين قيم مصفوفة الارتباك لكل الأصناف التي تم التعرف عليها

```
def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize=(12,12))

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
```

```
    plt.title(title, fontsize=18)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45, fontsize=8)
    plt.yticks(tick_marks, classes, fontsize=12)

    fmt = '.2f'
    thresh = cm.max() / 2.
    For I, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, I, format(cm[I, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[I, j] > thresh else "black")

    plt.ylabel('True label', fontsize=16)
    plt.xlabel('Predicted label', fontsize=16)
```

التعليمة التالية تستخدم من اجل استدعاء تابع الرسم السابق

```
plot_confusion_matrix(cnf_matrix, list(classes.values()))
```

التعليمة التالية تستخدم من اجل تقديم معلومات عن أداء النموذج الذي تم تدريبه على قاعدة البيانات،

```
print(classification_report(test_generator.classes, y_pred, target_names=list(classes.values())))
```

التابع التالي يستخدم من اجل قراءة صور من مسار معين

```
def load_image(filename):
```

```
    img = cv2.imread(os.path.join(BASE_DATASET_FOLDER, TEST_FOLDER, filen
ame))
    img = cv2.resize(img, (IMAGE_SIZE[0], IMAGE_SIZE[1]) )
    img = img /255

    return img
```

التابع التالي يستخدم من اجل قراءة الصور التي تم تحميلها وتقديمها الى النموذج (model) من اجل التعرف عليها

```
def predict(image):
    probabilities = model.predict(np.asarray([img]))[0]
    class_idx = np.argmax(probabilities)

    return {classes[class_idx]: probabilities[class_idx]}
```

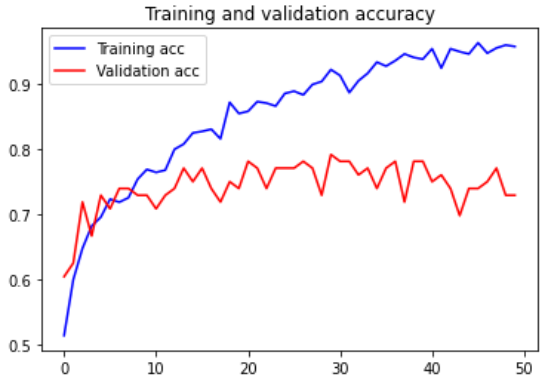
الكتلة البرمجية التالية تستخدم التابعين السابقين من اجل قراءة 10 صور عشوائية من قاعدة البيانات والتعرف عليها باستخدام الـ model الذي تم تدريبه

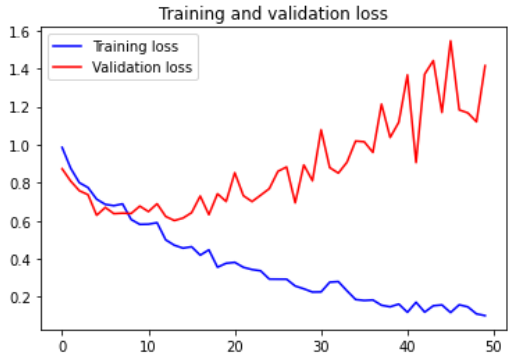
```
for idx, filename in enumerate(random.sample(test_generator.filenames, 10)):
    print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))

    img = load_image(filename)
    prediction = predict(img)
```

```
        print("PREDICTED: class: %s, confidence: %f" % (list(prediction.keys())[0], list(pr
ediction.values())[0]))
        plt.imshow(img)
        plt.figure(idx)
        plt.show()
```

5- النتائج والمناقشة

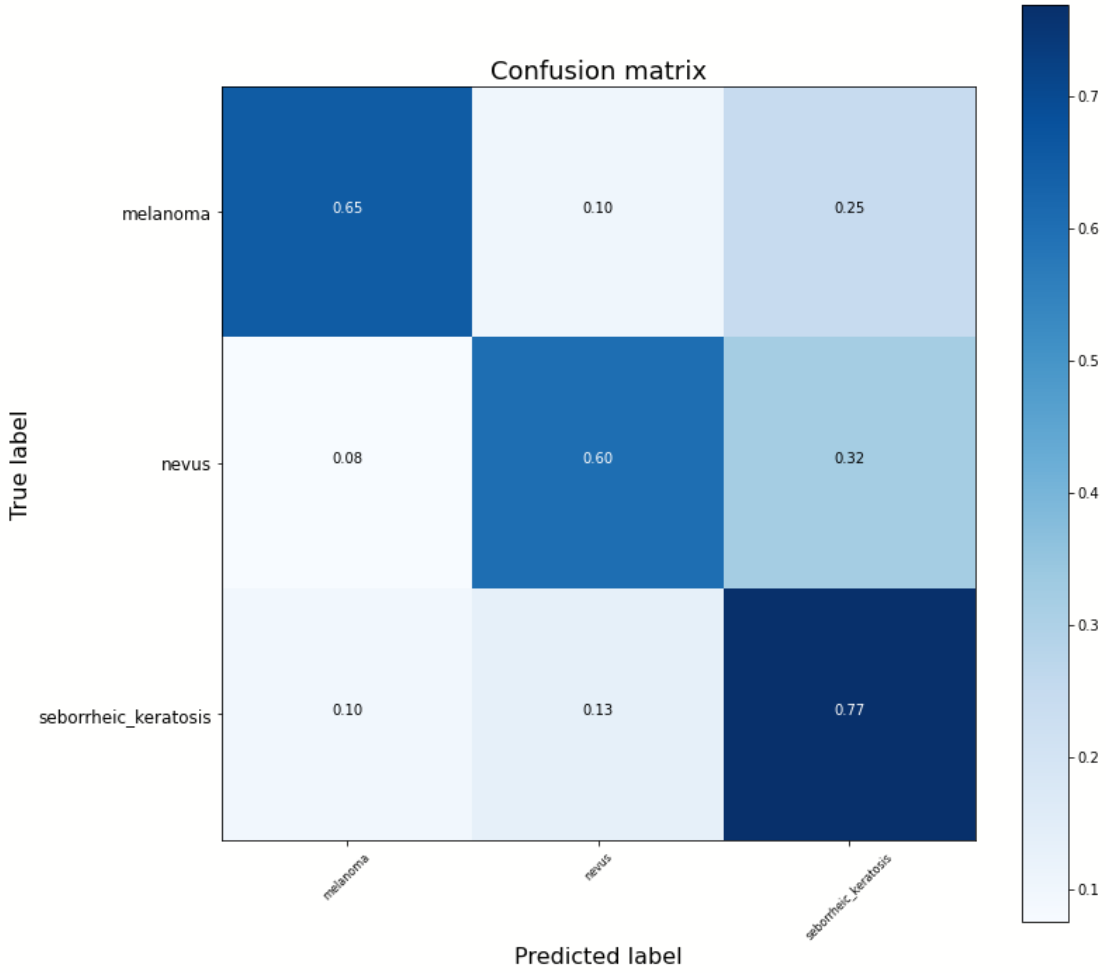
الشكل التالي يوضح منحنيات الدقة والخسارة خلال عملية التدريب.





وصلت دقة النموذج المستخدم الى 70%

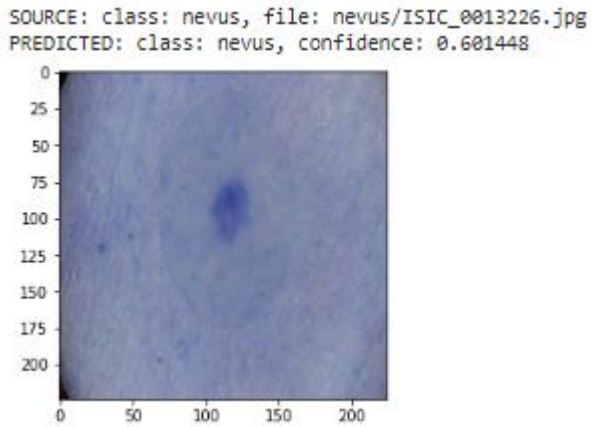
والشكل التالي يوضح قيم مصفوفة الارتباك لكل الأصناف الثلاثة المستخدمة:



نتائج اختبار النموذج المستخدم على عينات الاختبار:

هذه العينة كانت nevus وتعرف عليها النموذج

على انها nevus بنسبة 60%



هذه العينة كانت nevus وتعرف عليها النموذج على انها

nevus بنسبة 69%

```
<Figure size 432x288 with 0 Axes>
SOURCE: class: nevus, file: nevus/ISIC_0012741.jpg
PREDICTED: class: nevus, confidence: 0.696138
```

هذه العينة كانت

seborrheic_keratosis

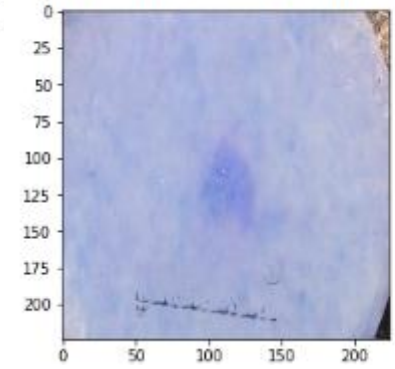
اخطأ النموذج في التعرف

عليها حيث تعرف عليها

على انها nevus بنسبة

%62

```
SOURCE: class: seborrheic_keratosis, file: seborrheic_keratosis/ISIC_0014526.jpg
PREDICTED: class: nevus, confidence: 0.625266
```



6- المراجع

[1] Qassim, H., Verma, A., & Feinzimer, D. (2018, January). Compressed residual-VGG16 CNN model for big data places image recognition. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 169-175). IEEE.

[2] Theckedath, D., & Sedamkar, R. R. (2020). Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. *SN Computer Science*, *1*(2), 1-7.

[3] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[4] Learning, D. (2020). Deep learning. *High-Dimensional Fuzzy Clustering*.

[5] https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c [accessed at: May, 28, 2022]

[6] https://www.kaggle.com/code/paoloripamonti/derma-diseases-detection/data