Damascus University

Computer Engineering

Computer Vision

Font Type Recognition

اعداد الطلاب

كنده أسعد صقر

محمد نور الدين قطليش

مقدمة

يعتبر اختيار نوع الخط من قبل المصمين هو أساس عملية التصميم.

قد يرى المصمم في كثير من الاحيان خطوط جديدة (في اللافتات الاعلانية, مواقع التوصل الاجتماعي, ..) وقد لا يستطيع التعرف على نوع هذا الخط.

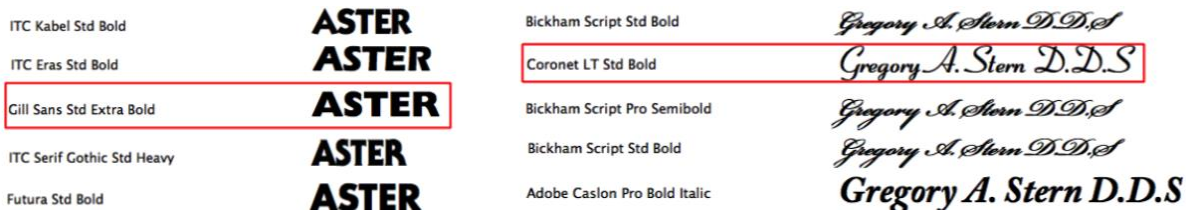
في العقد الاخير تم بناء قاعدة بيانات من قبل شركة ***Adobe*** لاكثر الخطوط استخداما من قبل المصممين.

مع تطور تقنيات التعلم العميق والتعرف على الاغراض, يمكن تطوير تقنية تمكننا من التعرف على نوع الخط ولغة الخط باستخدام الشبكات العصبونية الالتفافية ***CNN*** مع استخدام تقنية (***VFR***) ***Visual Font Recognition*** .

باستخام نماذج التعلم العميق الجاهزة مثل $(ResNet, ImageNet, GoogleNet)$ سيكون من الصعب تشكيل نموذج قادر على تصنيف الخطوط بسبب تحديات كبيرة في نوع البيانات المستخدم والتشابه الكبير في الخطوط والعينات $grayscale$.

DataSet

قمنا باستخدام مجموعة صغيرة من $AdobeVFR\ Dataset$ التي تعتبر اكبر قاعدة بيانات للتصاميم المكتوبة والخطوط قد تصل ل 200 ألف عينة.



معالجة الصورة قبل عملية التدريب

ليس كل البيانات والعينات تكون جاهزة للتدريب, من الضروري ان تتم عملية معالج وتحويل للصور المدخلة للنموذج قبل عملية التدريب, لتجنب مشكلة ***overfitting*** وهي مشكلة تظهر عند تدريب عينات شبه مثالية فتصبح الدقة في التدريب عالية جدا لكن الدقة في التنفيذ منعدمة, بعض الاوراق البحثية اظهرت ان عملية "**ما قبل العالجة**" التي تتم على العينات هي اساس تقديم نماذج ذات دقة عالية.

ومن اهم هذه العمليات والتحولات التي تجري:

- ***Noise***: وهي تطبيق ضجيج غاوصي صغير مع تغير شبه معتدل في الانحراف والمتوسط يضافو للدخل. باستخدام مكتبة ***PIL*** قمنا بانشاء التابع التالي:

```
def noise_image(pil_im):

    img_array = np.asarray(pil_im)
    #mean     # some constant
    #std    # standard deviation

    noisy_img = img_array + np.random.normal(mean = 0.1, std = 5 , img_array.shape)
    noisy_img_clipped = np.clip(noisy_img, 0, 255)
    noise_img = PIL.Image.fromarray(np.uint8(noisy_img_clipped)) # output
    plt.imshow((noisy_img_clipped ).astype(np.uint8)) # showing the image
    noise_img=noise_img.resize((105,105))
    return noise_img
```

- ***Blur***: وهي تطبيق ضبابية صغيرة من غاوص مع تغير بالانحراف بين 2.5 و 3.5

```
def blur_image(pil_im):

    blur_img = pil_im.filter(ImageFilter.GaussianBlur(radius=3)) # ouput
    blur_img=blur_img.resize((105,105))
    plt.imshow(blur_img)

    return blur_img
```

- ***Prespective Rotation***: نقوم بعملية تدوير عشوائي بشكل منطقي " اي نسب صغيرة" مما يجعل عملية التعلم تكون احترافية أكثر.

باستخدام مكتبات ***CV2, PIL*** قمنا بانشاء التابع التالي:

```
def affine_rotation(img):
    #the dimention of the image
    rows, columns = img.shape

    # rotation point for the affine function
    point1 = np.float32([[10, 10], [30, 10], [10, 30]])
    point2 = np.float32([[20, 15], [40, 10], [20, 40]])

    #the rotation methods
    A = cv2.getAffineTransform(point1, point2)
    output = cv2.warpAffine(img, A, (columns, rows))
    affine_img = PIL.Image.fromarray(np.uint8(output)) # affine rotated output

    # showing the imageZ
    plt.imshow(output)
    affine_img=affine_img.resize((105,105))


    return affine_img
```

Laplacian of gaussian: عملية كشف للحواف في مناطق معدومة وعميقة وشديدة الاختلاف

باستخدام مكتبات ***CV2*** قمنا بانشاء التابع التالي:

```
def gradient_fill(image):

    image=cv2.imread(img_path,0)
    laplacian = cv2.Laplacian(image,cv2.CV_64F)
    laplacian = cv2.resize(laplacian, (105, 105))
    return laplacian
```

$CNN\ Model$

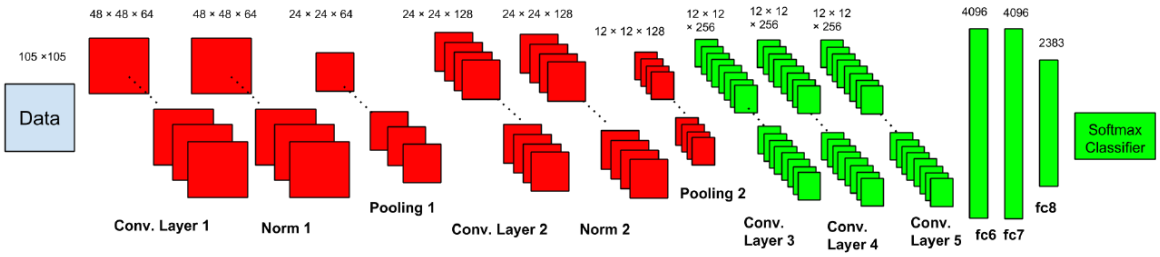
ستعتمد عملية التعليم على شبكة CNN متعددة الطبقات, حيث:

- في **الطبقات الدنيا** سوف يكون هناك عملية تعليم للميزات الصغيرة, تمييز الحواف والانحنائات والزوايا.
- في **الطبقات العليا** سوف تكون هناك شبكة تصنيف تعتمد على نتائج الطبقات الدنيا.

هناك الكثير من المعماريات والنماذج التي ممكن ان يستفاد منها في عملية التعليم, حيث في الكثير من الاوراق البحثية اعتمدت المصنفات على شبكات ال $ImageNet$ التي تعتبر الشبكة الام لكثير من الشبكات الاخرى.

تتبع المتغيرات والمدخلات خلال الشبكة سيحدد الخرج المطلوب وذلك اعتماد على الخوارزميات التي ستعتمدها هذه الطبقات, حيث حجم شعاع الدخل $(105, 105)$ وهي عينات تمت تسويتها للدخول للشبكة.

وتتكون الشبكة من ما يلي من الطبقات والفلاتر والتحولات:



تم تشكل هذه الشبكة عبر بيئة $TensorFlow$ المقدمة من شركة $Google$ حيث قمنا بانشاء نموذج يستقبل العينات في الدخل بشعاع $(105, 105, 1)$, العمق أخذ القيمة 1 لأن العينات ليست $RGB\ Color$.

يكون تطبيق النموذج كما يلي:

```
def create_model():
    model=Sequential()

    # low level Layers | الطبقات الدنيا
    model.add(Conv2D(64,
                     kernel_size=(48, 48),
                     activation='relu',
                     padding='same' ,
                     input_shape=(105,105,1)))

    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2),
                           padding='same'))

    model.add(Conv2D(128, kernel_size=(24, 24),
                     padding='same',
                     activation='relu'))

    model.add(BatchNormalization())
    model.add(MaxPooling2D(pool_size=(2, 2),
                           padding='same'))

    model.add(Conv2DTranspose(128, (24,24),
                              strides = (1,1),
                              activation = 'relu',
                              padding='same',
                              kernel_initializer='uniform'))
    model.add(UpSampling2D(size=(2, 2)))

    model.add(Conv2DTranspose(64, (12,12),
                              strides = (1,1),
                              activation = 'relu',
                              padding='same',
                              kernel_initializer='uniform'))
    model.add(UpSampling2D(size=(2, 2)))

    #high level layers | الطبقات العليا
    model.add(Conv2D(256, kernel_size=(12, 12), activation='relu',padding='same'))

    model.add(Conv2D(256, kernel_size=(12, 12), activation='relu',padding='same'))

    model.add(Conv2D(256, kernel_size=(12, 12), activation='relu',padding='same'))

    # توحيد جميع المبوات في شعاع وحيد للتعرف عليها
    model.add(Flatten())

    model.add(Dense(4096, activation='relu'))

    model.add(Dropout(0.5))

    model.add(Dense(4096,activation='relu'))

    model.add(Dropout(0.5))

    model.add(Dense(2383,activation='relu'))

    model.add(Dense(5, activation='softmax'))

    return model
```

Training the Model

بسبب صعوبة المعالجة والتدريب في الحواسيب الشخصية بسبب قلة ذاكرة المعالجة الموجودة في وحدات معالجة الصورة ***GPU*** لذلك استعنا بمنصة ***Google colab*** التي تقدم وحدات معالجة فائقة السرعة لعملية التدريب وكانت النتائج في أول عملية تدريب:

```
model.fit(trainX, trainY,shuffle=True,
          batch_size=batch_size,
          epochs=50,
          verbose=1,
          validation_data=(testX, testY))
```

```
Epoch 1/50
8/8 [==============================] - 39s 3s/step - loss: 0.1593 - accuracy: 0.2778 - val_loss: 0.1588 - val_accuracy: 0.4167
Epoch 2/50
8/8 [==============================] - 14s 2s/step - loss: 0.1564 - accuracy: 0.3526 - val_loss: 0.1587 - val_accuracy: 0.4167
Epoch 3/50
8/8 [==============================] - 14s 2s/step - loss: 0.1562 - accuracy: 0.3141 - val_loss: 0.1562 - val_accuracy: 0.4167
Epoch 4/50
8/8 [==============================] - 14s 2s/step - loss: 0.1539 - accuracy: 0.3739 - val_loss: 0.1568 - val_accuracy: 0.4167
Epoch 5/50
8/8 [==============================] - 14s 2s/step - loss: 0.1532 - accuracy: 0.3739 - val_loss: 0.1565 - val_accuracy: 0.4167
Epoch 6/50
8/8 [==============================] - 14s 2s/step - loss: 0.1535 - accuracy: 0.3739 - val_loss: 0.1539 - val_accuracy: 0.4167
Epoch 7/50
8/8 [==============================] - 14s 2s/step - loss: 0.1524 - accuracy: 0.3739 - val_loss: 0.1534 - val_accuracy: 0.4167
Epoch 8/50
8/8 [==============================] - 14s 2s/step - loss: 0.1522 - accuracy: 0.3739 - val_loss: 0.1543 - val_accuracy: 0.4167
Epoch 9/50
8/8 [==============================] - 14s 2s/step - loss: 0.1529 - accuracy: 0.3739 - val_loss: 0.1516 - val_accuracy: 0.4167
Epoch 10/50
8/8 [==============================] - 14s 2s/step - loss: 0.1533 - accuracy: 0.3739 - val_loss: 0.1514 - val_accuracy: 0.4167
Epoch 11/50
8/8 [==============================] - 14s 2s/step - loss: 0.1505 - accuracy: 0.3739 - val_loss: 0.1504 - val_accuracy: 0.4167
```

- كانت الدقة في البداية ضعيفة جدا واصبح الخطأ في فترات محددة يرتفع دليل على تنوع البيانات والكبير وانها ليست من نفس التصنيق,

النتائج النهائية لعملية التدريب

```
Epoch 12/50
8/8 [==============================] - 14s 2s/step - loss: 0.0354 - accuracy: 0.8718 - val_loss: 0.0547 - val_accuracy: 0.8462
Epoch 13/50
8/8 [==============================] - 14s 2s/step - loss: 0.0383 - accuracy: 0.8526 - val_loss: 0.0527 - val_accuracy: 0.8462
Epoch 14/50
8/8 [==============================] - 14s 2s/step - loss: 0.0352 - accuracy: 0.8697 - val_loss: 0.0632 - val_accuracy: 0.7692
Epoch 15/50
8/8 [==============================] - 14s 2s/step - loss: 0.0319 - accuracy: 0.8974 - val_loss: 0.0883 - val_accuracy: 0.7244
Epoch 16/50
8/8 [==============================] - 14s 2s/step - loss: 0.0292 - accuracy: 0.8974 - val_loss: 0.0895 - val_accuracy: 0.6731
Epoch 17/50
8/8 [==============================] - 14s 2s/step - loss: 0.0346 - accuracy: 0.8547 - val_loss: 0.0886 - val_accuracy: 0.6731
Epoch 18/50
8/8 [==============================] - 14s 2s/step - loss: 0.0285 - accuracy: 0.8974 - val_loss: 0.0730 - val_accuracy: 0.7821
Epoch 19/50
8/8 [==============================] - 14s 2s/step - loss: 0.0283 - accuracy: 0.9103 - val_loss: 0.0485 - val_accuracy: 0.8333
Epoch 20/50
8/8 [==============================] - 14s 2s/step - loss: 0.0300 - accuracy: 0.8953 - val_loss: 0.0428 - val_accuracy: 0.8590
Epoch 21/50
```