Outline

Deep Learning

Midterm Assignment

P.L.D. Tien (520K0220)

Ton Duc Thang University

March 11, 2023

Outline

Introduction

Overview

ResNet50

ResNet50 (Overfit)

ResNet50 (Vanishing)

CNN

CNN (Overfit)

CNN (Vanishing)

Overall results

End

What is it?

An image captioning system, using ResNet50 and CNN, with independent handling of overfitting and vanishing problems. This submission uses the CIFAR-10[1] for it's dataset.

Note

The snippets of code are coming from the original notebook. It's advised to look at it for a better understanding.

[1] https://www.cs.toronto.edu/ kriz/cifar-10-python.tar.gz

What does it cover?

- ResNet50
- ResNet50 (w/ Overfit handling)
- ResNet50 (w/ Vanishing handling)
- CNN
- CNN (w/ Overfit handling)
- CNN (w/ Vanishing handling)

Implementation

Intentionally left blank.

Too much to cover here, read the notebook.

Results

a

Overview

```
X = AveragePooling2D(pool_size=(2, 2), padding='same')(X)
X = Flatten()(X)
#X = Dense(256, activation='relu', name='fc1', kernel_initializer='glorot_uniform')(X)
#X = Dense(128, activation='relu', name='fc2', kernel_initializer='glorot_uniform')(X)

X = Dense(256, activation='relu', name='fc1', kernel_initializer='glorot_uniform'\
          , kernel_regularizer=regularizers.l2(0.01))(X)
X = Dense(128, activation='relu', name='fc2', kernel_initializer='glorot_uniform'\
          , kernel_regularizer=regularizers.l2(0.01))(X)
```

Replace these lines with regularizers. Dropout doesn't work here since they're not tensors.

Results

a

Implementation


```
# add batch normalization after ReLU activation
X = convolutional_block(X, f=3, filters=[64, 64, 256], stage=2, block='a', stride=1)
X = BatchNormalization(axis=3, name='bn_conv2')(X)
X = ReLU()(X)
X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
X = BatchNormalization(axis=3, name='bn_conv3')(X)
X = ReLU()(X)
X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')
X = BatchNormalization(axis=3, name='bn_conv4')(X)
X = ReLU()(X)

X = convolutional_block(X, f=3, filters=[128, 128, 512], stage=3, block='a', stride=2)
X = BatchNormalization(axis=3, name='bn_conv5')(X)
X = ReLU()(X)
X = identity_block(X, 3, [128, 128, 512], stage=3, block='b')
X = BatchNormalization(axis=3, name='bn_conv6')(X)
X = ReLU()(X)
X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
X = BatchNormalization(axis=3, name='bn_conv7')(X)
X = ReLU()(X)
X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')
```

We add BatchNormalizatio after ReLU.

Results

a

Overview


```
model = models.Sequential(
[

    layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32,
    layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),

    layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),

    layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),

    layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
    layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),

    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.BatchNormalization(),

    layers.Dense(10, activation='softmax')
])
```

Simple enough.

Results

a

Overview

```
layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input
layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.1),

layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.1),

layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.1),

layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'),
layers.BatchNormalization(),
layers.MaxPooling2D((2, 2)),
layers.Dropout(0.1),
```

We simply add Dropout in between the blocks.

Results

a

Implementation


```
2
3   model = models.Sequential(
4   [
5       layers.Conv2D(32, (3, 3), activation='r
6       layers.BatchNormalization(),
7       layers.Conv2D(32, (3, 3), activation='r
8       layers.BatchNormalization(),
9       layers.MaxPooling2D((2, 2)),
10      layers.Dropout(0.2),
11
12
13      layers.Conv2D(64, (3, 3), activation='r
14      layers.BatchNormalization(),
15      layers.Conv2D(64, (3, 3), activation='r
16      layers.BatchNormalization(),
17      layers.Conv2D(64, (3, 3), activation='r
18      layers.BatchNormalization(),
19      layers.MaxPooling2D((2, 2)),
20      layers.Dropout(0.3),
21
22
```

Add BatchNormalizatio in between Conv2D.

Results

a

All six of them together

a

Anything to talk about?