Austin Vornhagen

9/20/2021

CSCI 8110

Assignment 1

Runnable codes:

```
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
import numpy as np
from tensorflow.keras import datasets
import matplotlib.pyplot as plt


(x_train, _), (x_test, _) = datasets.cifar10.load_data()

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0
x_train = np.reshape(x_train, (len(x_train), 32, 32, 3))
x_test = np.reshape(x_test, (len(x_test), 32, 32, 3))

noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal (loc=0.0, scale=
1.0, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0
, size=x_test.shape)

x_train_noisy = np.clip(x_train_noisy, 0., 1.)
x_test_noisy = np.clip(x_test_noisy, 0., 1.)
input_img = Input(shape=(32,32,3))

x = Conv2D(32, (3,3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2,2), padding='same')(x)
x = Conv2D(32, (3,3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2,2), padding='same')(x)

x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
```

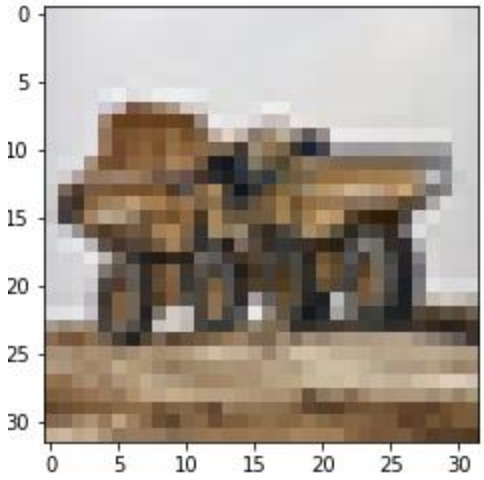
```
autoencoder.compile(optimizer='adam', loss='mae',metrics='accuracy'

)

autoencoder.summary()

autoencoder.fit(x_train_noisy, x_train, epochs=100, batch_size=128, shuffl
e=True, validation_data=(x_test_noisy, x_test))

prediction_imgs = autoencoder.predict(x_test_noisy)

test_loss, test_acc = autoencoder.evaluate(x_test_noisy,x_test, verbose=2)
print(test_acc)
print(test_loss)

plt.imshow(x_test[171] ,cmap=plt.cm.binary)
plt.show()
plt.imshow(x_test_noisy[171] ,cmap=plt.cm.binary)
plt.show()
plt.imshow(prediction_imgs[171] ,cmap=plt.cm.binary)
plt.show()
```

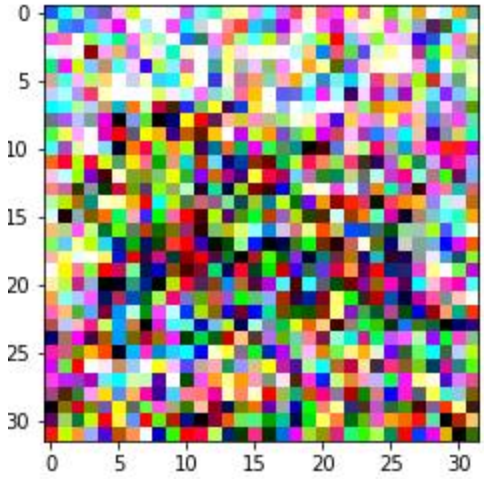
Plot Inputs and their corresponding outputs:

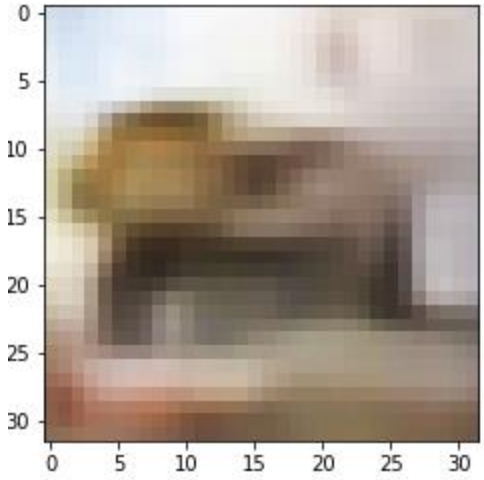
Input Image 171

Noise Image 171

Prediction Image 171



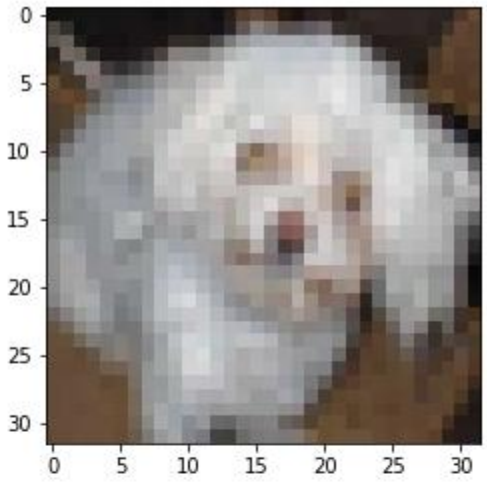


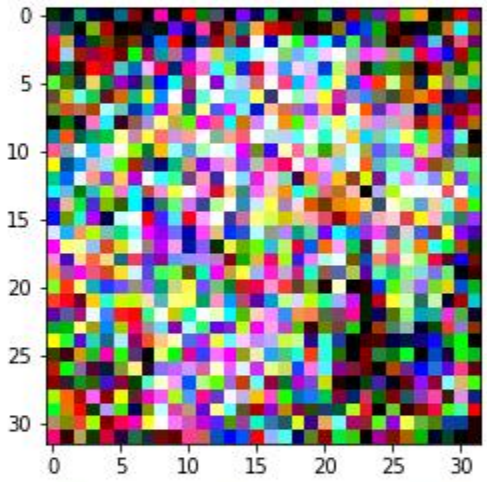


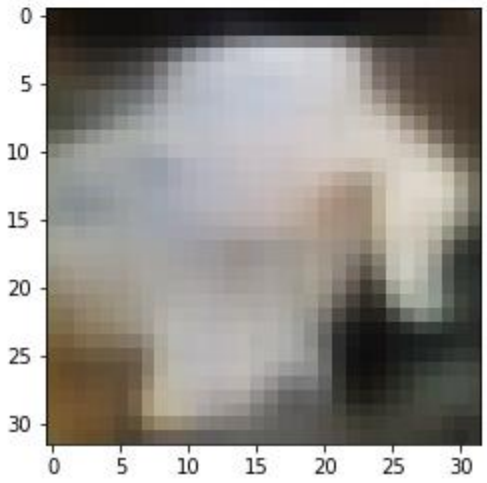
Input Image 1000

Noise Image 1000

Prediction Image 1000



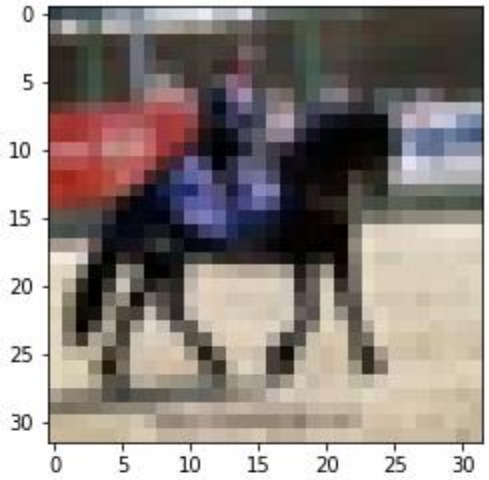


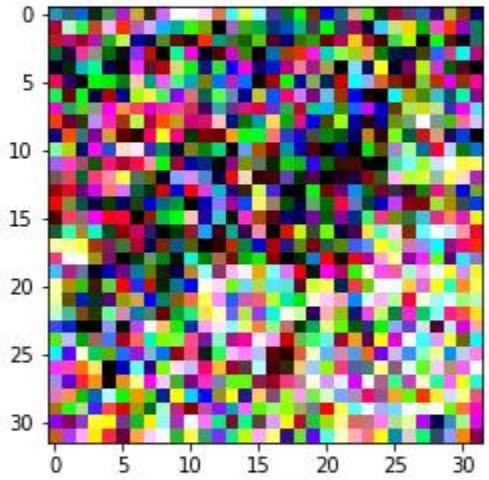


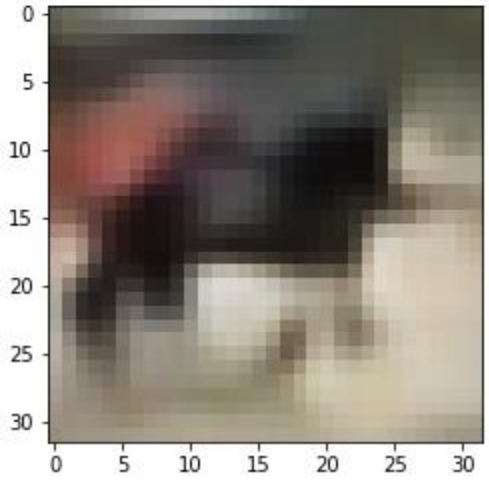
Input Image 5000

Noise Image 5000

Prediction Image 5000







Discussion of Thinking and Discovery:

This is my first time using deep learning syntax. I got the structure from lecture 4 but did not know any of the syntax when starting the project. The numpy reshape function adds dimensions to the array of input data. Conv2D is a 2D convolution layer. My understanding is this layer creates a matrix called a convolution kernel and this matrix is combined with the layer input to produce a tensor. MaxPooling2D is a layer that downsamples the input. UpSampling2D is a layer that upsamples the input. The Model class groups layers into an object with training features. Fit trains the model. Predict generates predictions based on the inputs.

I started the project with 100 epochs. The loss represents the value of the cost function for the training data, and the val_loss is the value of the cost function for the cross-validation data. The Accuracy is what was trained against and the val_accuracy is the accuracy of the validation set. I had trouble getting the evaluate method to work because I didn't know I needed to add the metrics='accuracy' parameter to the compile method. Below are the results of changes in the number of epochs:

Epochs	Loss	Val_Loss	Accuracy	Val_Accuracy
5	0.0746	0.0746	0.6730	0.6763
100	0.0741	0.0742	0.6749	0.6634
500	0.0735	0.0738	0.6753	0.6596

You can see that the more epochs we did, the better the model performed. However, the amount of time it takes to see improvement in the model increases exponentially as you have to really increase the number of epochs in order to get the same improvements.

The project started with a noise factor of 0.5. The noise factor is used to add "noise" to the images by changing the color value of the pixels. Below are the results of changing the noise factor with 100 epochs:

Noise Factor	Loss	Val_Loss	Accuracy	Val_Accuracy
0.5	0.0741	0.0742	0.6749	0.6634
0	0.0309	0.0306	0.8321	0.8355
1	0.1036	0.1036	0.1036	0.1036
2	0.1380	0.1395	0.5194	0.5271
10	0.1946	0.2024	0.4579	0.4513

You can see the more noise is present, the higher the loss goes and the lower the accuracy goes.

A summary of the original layers present in the model looks like this:

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
```

```
conv2d_5 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
up_sampling2d_3 (UpSampling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 32, 32, 3)         867
=================================================================
Total params: 29,507
Trainable params: 29,507
Non-trainable params: 0
```

From above, you can see there are 10 layers present. To see what happens when I change the layers, I can change my code to this:

```
x = Conv2D(32, (3,3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2,2), padding='same')(x)
x = Conv2D(32, (3,3), activation='relu', padding='same')(x)
x = MaxPooling2D((2,2), padding='same')(x)
x = Conv2D(32, (3,3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2,2), padding='same')(x)


x = Conv2D(32, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(x)
```

A summary of what the model looks like with these code changes:

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_3 (InputLayer)         [(None, 32, 32, 3)]       0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 32, 32, 32)        896
```

```
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 16, 16, 32)        9248
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 8, 8, 32)          9248
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 4, 4, 32)          0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 4, 4, 32)          9248
_________________________________________________________________
up_sampling2d_8 (UpSampling2 (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 8, 8, 32)          9248
_________________________________________________________________
up_sampling2d_9 (UpSampling2 (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_23 (Conv2D)           (None, 16, 16, 32)        9248
_________________________________________________________________
up_sampling2d_10 (UpSampling (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 32, 32, 3)         867
=================================================================
Total params: 48,003
Trainable params: 48,003
Non-trainable params: 0
```

From above, you can see that there are now 14 layers present in the model. When the new model is run with 0.5 noise factor and for 100 epochs we get:

loss: 0.0775 - accuracy: 0.6814 - val_loss: 0.0780 - val_accuracy: 0.6728

Adding layers decreased the loss and increased the accuracy of the model.

The last thing I found was that there is a huge difference in execution performance of the model when using a GPU vs not using a GPU. When performing 100 epochs on the data, it takes 3 minutes and 3 seconds per epoch without a GPU and 9 seconds with a GPU. This is a huge difference and something that makes a GPU very helpful when performing deep learning.