File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Gemini

```
[28]    input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS),
        kernel_size=(2, 2), # Changed kernel size to (2, 2) or (1, 1)
        filters=8,
        strides=1,
        activation=tf.keras.activations.relu,
        kernel_initializer=tf.keras.initializers.VarianceScaling(),
        padding='same'  # Add padding to avoid negative dimensions
    ))

    # ... (rest of the model definition)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 2, 2, 8)	72

Total params: 72 (288.00 B)
Trainable params: 72 (288.00 B)
Non-trainable params: 0 (0.00 B)

```
[ ] tf.keras.utils.plot_model(
        model,
        show_shapes=True,
```

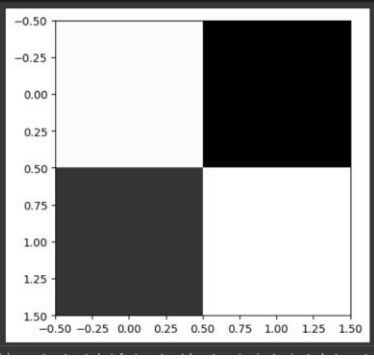
File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

RAM Disk Gemini

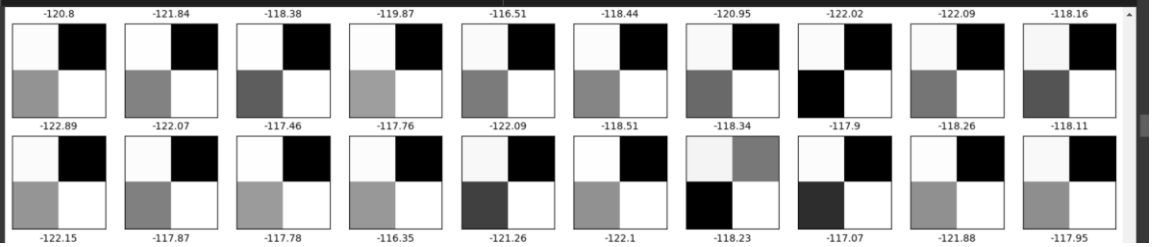
```
[22] # For grayscale, assuming the last two dimensions represent the image:
     plt.imshow(X_train[0, :, :, 0], cmap=plt.cm.binary)  # Take the first channel
     plt.show()

     # Alternatively, if you want to combine channels, you can sum or average them:
     # plt.imshow(X_train[0].sum(axis=-1), cmap=plt.cm.binary)  # Sum across channels
     # plt.imshow(X_train[0].mean(axis=-1), cmap=plt.cm.binary)  # Average across channels
```

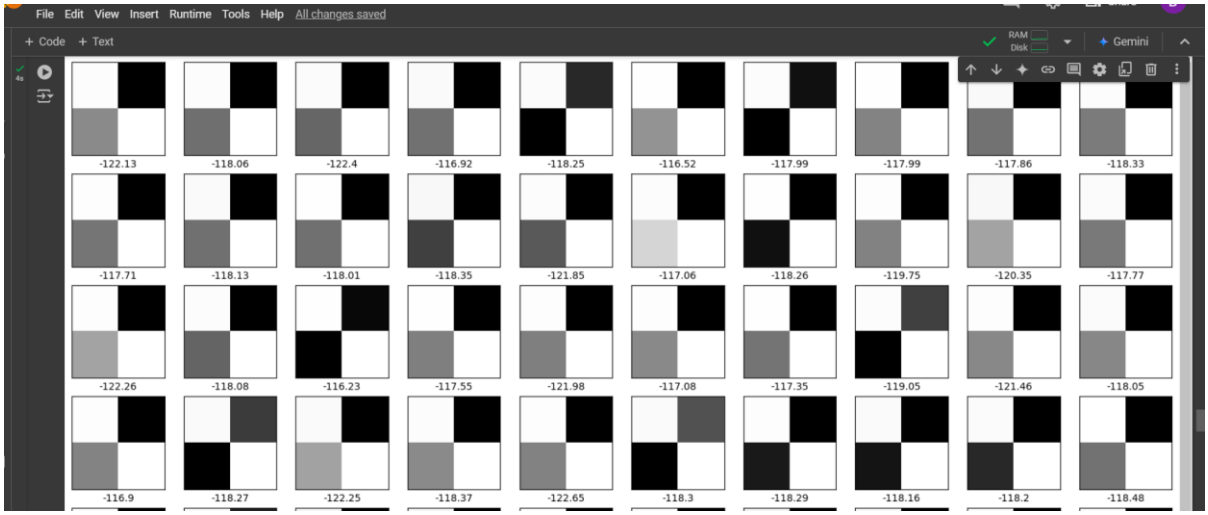


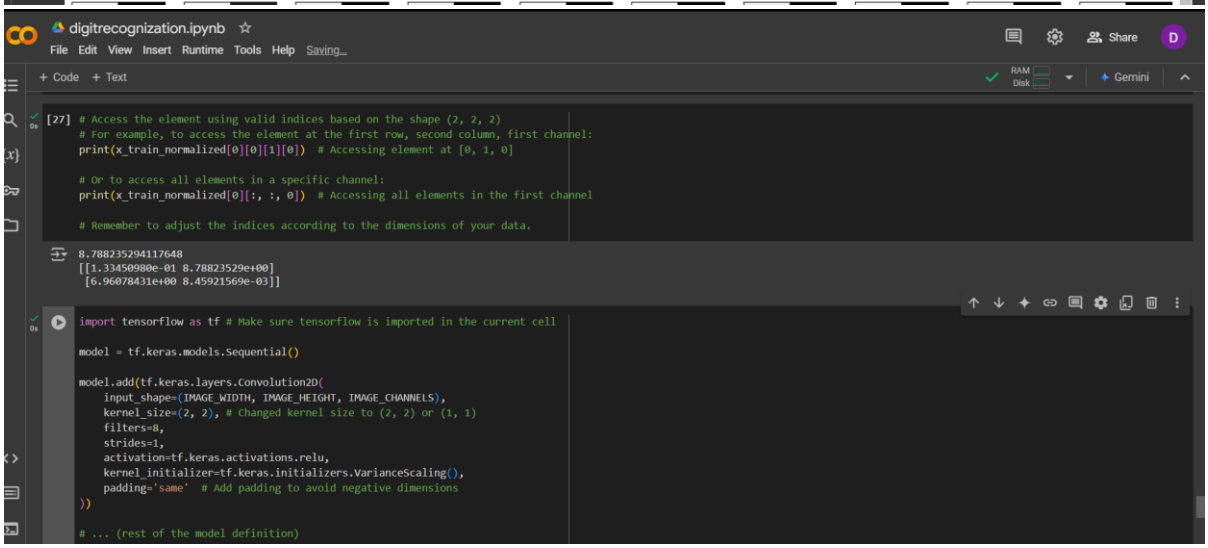
+ Code + Text

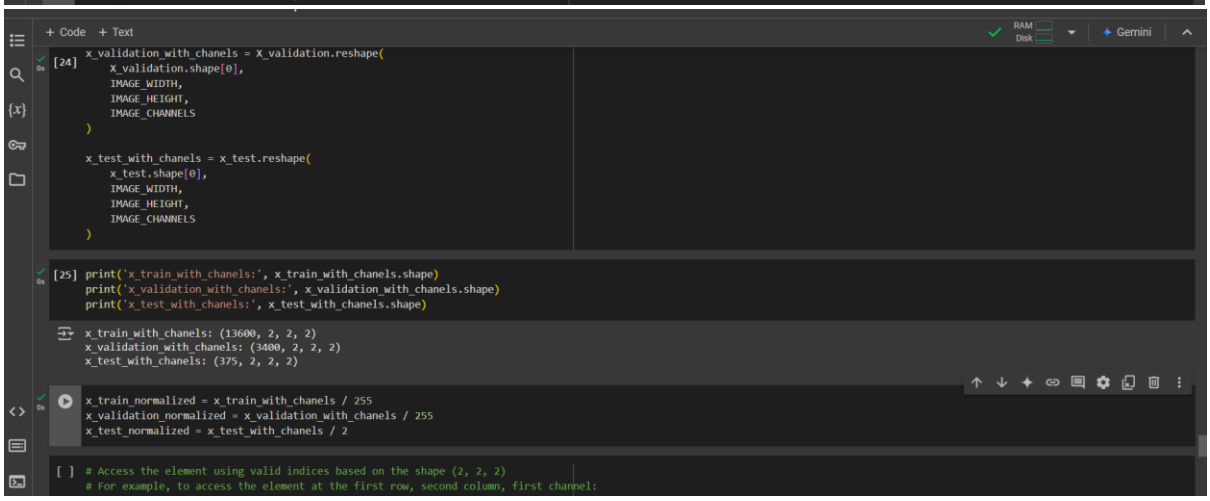
RAM Disk Gemini

```
import math
numbers_to_display = 100
num_cells = math.ceil(math.sqrt(numbers_to_display))
plt.figure(figsize=(20,20))
for i in range(numbers_to_display):
    plt.subplot(num_cells, num_cells, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    # Assuming X_train contains the image data
    plt.imshow(X_train[i, :, :, 0], cmap=plt.cm.binary)  # Accessing the first channel for grayscale
    # If y_train contains labels corresponding to X_train
    plt.xlabel(y_train[i])
plt.show()
```




File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM
Disk
Gemini
-122.13 -118.06 -122.4 -116.92 -118.25 -116.52 -117.99 -117.99 -117.86 -118.33
-117.71 -118.13 -118.01 -118.35 -121.85 -117.06 -118.26 -119.75 -120.35 -117.77
-122.26 -118.08 -116.23 -117.55 -121.98 -117.08 -117.35 -119.05 -121.46 -118.05
-116.9 -118.27 -122.25 -118.37 -122.65 -118.3 -118.29 -118.16 -118.2 -118.48


digitrecognization.ipynb
File Edit View Insert Runtime Tools Help Saving...
Share
D
+ Code + Text
RAM
Disk
Gemini
[27] # Access the element using valid indices based on the shape (2, 2, 2)
For example, to access the element at the first row, second column, first channel:
print(x_train_normalized[0][0][1][0]) # Accessing element at [0, 1, 0]
Or to access all elements in a specific channel:
print(x_train_normalized[0][:, :, 0]) # Accessing all elements in the first channel
Remember to adjust the indices according to the dimensions of your data.
8.788235294117648
[[1.33450980e-01 8.78823529e+00]
[6.96078431e+00 8.45921569e-03]]
import tensorflow as tf # Make sure tensorflow is imported in the current cell
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Convolution2D(
input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS),
kernel_size=(2, 2), # Changed kernel size to (2, 2) or (1, 1)
filters=8,
strides=1,
activation=tf.keras.activations.relu,
kernel_initializer=tf.keras.initializers.VarianceScaling(),
padding='same' # Add padding to avoid negative dimensions
))
... (rest of the model definition)


+ Code + Text
RAM
Disk
Gemini
[24] x_validation_with_chanels = X_validation.reshape(
X_validation.shape[0],
IMAGE_WIDTH,
IMAGE_HEIGHT,
IMAGE_CHANNELS
)
x_test_with_chanels = x_test.reshape(
x_test.shape[0],
IMAGE_WIDTH,
IMAGE_HEIGHT,
IMAGE_CHANNELS
)
[25] print('x_train_with_chanels:', x_train_with_chanels.shape)
print('x_validation_with_chanels:', x_validation_with_chanels.shape)
print('x_test_with_chanels:', x_test_with_chanels.shape)
x_train_with_chanels: (13600, 2, 2, 2)
x_validation_with_chanels: (3400, 2, 2, 2)
x_test_with_chanels: (375, 2, 2, 2)
x_train_normalized = x_train_with_chanels / 255
x_validation_normalized = x_validation_with_chanels / 255
x_test_normalized = x_test_with_chanels / 2
[ ] # Access the element using valid indices based on the shape (2, 2, 2)
For example, to access the element at the first row, second column, first channel:

digitrecognization.ipynb

File Edit View Insert Runtime Tools Help

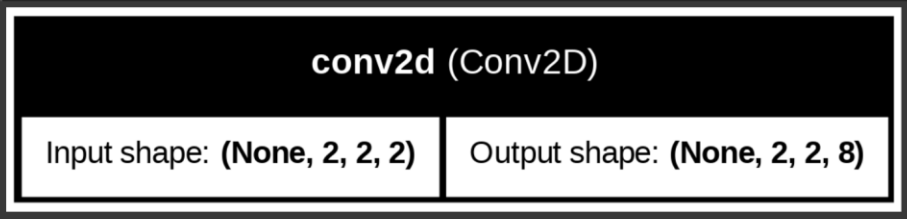
Share

+ Code + Text

Gemini

```
[29]  Total params: 72 (288.00 B)
      Trainable params: 72 (288.00 B)
      Non-trainable params: 0 (0.00 B)
```

```
tf.keras.utils.plot_model(
    model,
    show_shapes=True,
    show_layer_names=True,
)
```



digitrecognization.ipynb - Colab

New Issue · diikshasharmaa/diksha

colab.research.google.com/drive/1sJtA4PyRb8VLI9bQRvCU_kbeEvhUuhun#scrollTo=FBBh6Gn4IIoz

This notebook is open with private outputs. Outputs will not be saved. You can disable this in Notebook settings.

digitrecognization.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Share

+ Code + Text

Gemini

```
[16]  X_train, X_validation, y_train, y_validation = train_test_split(X, y, test_size = 0.2,random_state = 1212)
```

```
print('X_train:', X_train.shape)
print('y_train:', y_train.shape)
print('X_validation:', X_validation.shape)
print('y_validation:', y_validation.shape)
```

```
X_train: (13600, 8)
y_train: (13600,)
X_validation: (3400, 8)
y_validation: (3400,)
```

```
[ ]  print('x_train:', X_train.shape)
     print('y_train:', y_train.shape)
     print('x_validation:', X_validation.shape)
     print('y_validation:', y_validation.shape)
     print('x_test:', X_test.shape)
```

```
[ ]  X_train = X_train.to_numpy().reshape(-1, 2, 2, 2)  # Reshape to dimensions compatible with the data
     y_train = y_train.values
     X_validation = X_validation.to_numpy().reshape(-1, 2, 2, 2)  # Reshape to dimensions compatible with the data
     y_validation = y_validation.values
     x_test = test.drop(columns=['longitude', 'latitude', 'housing_median_age', 'total_rooms', 'total_bedrooms', 'population', 'households', 'median_income']).to_numpy().reshape(-1, 2, 2
```

```
[ ]  # Save image parameters to the constants that we will use later for data re-shaping and for model training.
```

0s completed at 19:21

18°C Haze

Search

ENG IN

19:22 14-12-2024