Tanawin-st123975-alexnet-cifar100

October 30, 2023

[15]:
```
!nvidia-smi
```

```
Mon Oct 30 04:20:53 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.105.17   Driver Version: 525.105.17   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   61C    P0    28W /  70W |   2987MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

[16]:
```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,␣
↪MaxPooling2D
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import LearningRateScheduler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras import regularizers
from tensorflow.keras.optimizers import SGD
```

[17]:
```
# Check if GPU is available
print("Num GPUs Available: ", len(tf.config.experimental.
↪list_physical_devices('GPU')))
```

```
# Check if Keras is using GPU
print("Keras GPU Available: ", tf.test.is_built_with_cuda())
```

```
Num GPUs Available:  1
Keras GPU Available:  True
```

[18]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Load CIFAR-100 data
(train_images, train_labels), (test_images, test_labels) = datasets.cifar100.
 ↪load_data()

# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0

# Apply data augmentation to the training data
datagen = ImageDataGenerator(
    rotation_range=40,  # Degree range for random rotations
    width_shift_range=0.2,  # Range for random horizontal shifts
    height_shift_range=0.2,  # Range for random vertical shifts
    shear_range=0.2,  # Shear Intensity
    zoom_range=0.2,  # Range for random zoom
    horizontal_flip=True,  # Randomly flip inputs horizontally
    fill_mode='nearest'  # Strategy for filling in newly created pixels
)

datagen.fit(train_images)
```

[19]:
```
def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * tf.math.exp(-0.1)
```

[20]:
```
from tensorflow.keras import models
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
 ↪Dropout, BatchNormalization
from tensorflow.keras import regularizers

# Initialize the model
model = models.Sequential()

# Add layers with increased regularization and batch normalization
model.add(Conv2D(64, (3, 3), padding='same', activation='relu',␣
 ↪input_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
```

```
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.
  ↪001)))
model.add(Dropout(0.5))
model.add(BatchNormalization())

model.add(Dense(512, activation='relu', kernel_regularizer=regularizers.l2(0.
  ↪001)))
model.add(Dropout(0.5))
model.add(BatchNormalization())

model.add(Dense(100, activation='softmax'))


# Compile the model with a lower learning rate and batch normalization
opt = Adam(learning_rate=0.001)
model.compile(optimizer=opt,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])


# # Compile the model with a lower learning rate and batch normalization
# opt = SGD(learning_rate=0.01, momentum=0.9)  # Set your desired learning rate␣
  ↪and momentum
# model.compile(optimizer=opt,
#               loss='sparse_categorical_crossentropy',
```

```
#                metrics=['accuracy'])
```

[21]:
```
from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
```

[22]:
```
batch_size = 64  # Set your desired batch size

# Train the model with the specified batch size
history = model.fit(train_images, train_labels,
                    batch_size=batch_size,
                    epochs=50,
                    validation_data=(test_images, test_labels),
                    callbacks=[early_stopping])


model.save('my_cifar100_model.h5')
```

```
Epoch 1/50
782/782 [==============================] - 28s 28ms/step - loss: 5.2554 -
accuracy: 0.0822 - val_loss: 4.4392 - val_accuracy: 0.1470
Epoch 2/50
782/782 [==============================] - 21s 27ms/step - loss: 4.0299 -
accuracy: 0.1805 - val_loss: 4.1300 - val_accuracy: 0.1904
Epoch 3/50
782/782 [==============================] - 21s 26ms/step - loss: 3.4609 -
accuracy: 0.2675 - val_loss: 3.7548 - val_accuracy: 0.2515
Epoch 4/50
782/782 [==============================] - 21s 27ms/step - loss: 3.1490 -
accuracy: 0.3360 - val_loss: 3.1101 - val_accuracy: 0.3580
Epoch 5/50
782/782 [==============================] - 21s 27ms/step - loss: 2.9304 -
accuracy: 0.3959 - val_loss: 2.7861 - val_accuracy: 0.4306
Epoch 6/50
782/782 [==============================] - 21s 27ms/step - loss: 2.7538 -
accuracy: 0.4445 - val_loss: 2.7502 - val_accuracy: 0.4627
Epoch 7/50
782/782 [==============================] - 21s 27ms/step - loss: 2.5787 -
accuracy: 0.4892 - val_loss: 2.6522 - val_accuracy: 0.4776
Epoch 8/50
782/782 [==============================] - 21s 26ms/step - loss: 2.4354 -
accuracy: 0.5317 - val_loss: 2.6823 - val_accuracy: 0.4905
Epoch 9/50
782/782 [==============================] - 21s 26ms/step - loss: 2.2994 -
accuracy: 0.5709 - val_loss: 2.6105 - val_accuracy: 0.5226
Epoch 10/50
```

```
782/782 [==============================] - 21s 27ms/step - loss: 2.1482 -
accuracy: 0.6149 - val_loss: 2.5657 - val_accuracy: 0.5354
Epoch 11/50
782/782 [==============================] - 21s 27ms/step - loss: 2.0215 -
accuracy: 0.6481 - val_loss: 2.5602 - val_accuracy: 0.5455
Epoch 12/50
782/782 [==============================] - 21s 27ms/step - loss: 1.9278 -
accuracy: 0.6792 - val_loss: 2.6559 - val_accuracy: 0.5387
Epoch 13/50
782/782 [==============================] - 21s 27ms/step - loss: 1.8172 -
accuracy: 0.7101 - val_loss: 2.6752 - val_accuracy: 0.5459
Epoch 14/50
782/782 [==============================] - 22s 28ms/step - loss: 1.7270 -
accuracy: 0.7351 - val_loss: 2.7542 - val_accuracy: 0.5416
Epoch 15/50
782/782 [==============================] - 21s 27ms/step - loss: 1.6359 -
accuracy: 0.7632 - val_loss: 2.7685 - val_accuracy: 0.5422
Epoch 16/50
782/782 [==============================] - 21s 27ms/step - loss: 1.6039 -
accuracy: 0.7757 - val_loss: 2.7748 - val_accuracy: 0.5500
```

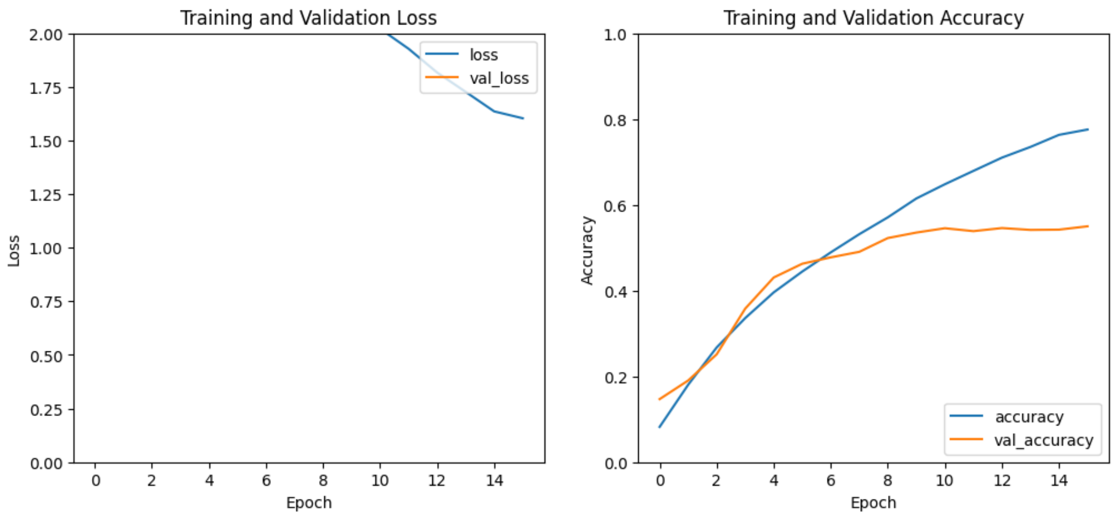
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079:
UserWarning: You are saving your model as an HDF5 file via `model.save()`. This
file format is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

[23]:

```
# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print('\nTest accuracy:', test_acc)

# Plot the training and validation accuracy and loss
plt.figure(figsize=(12, 5))

# Plot training and validation accuracy
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')

# Plot training and validation loss
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
```

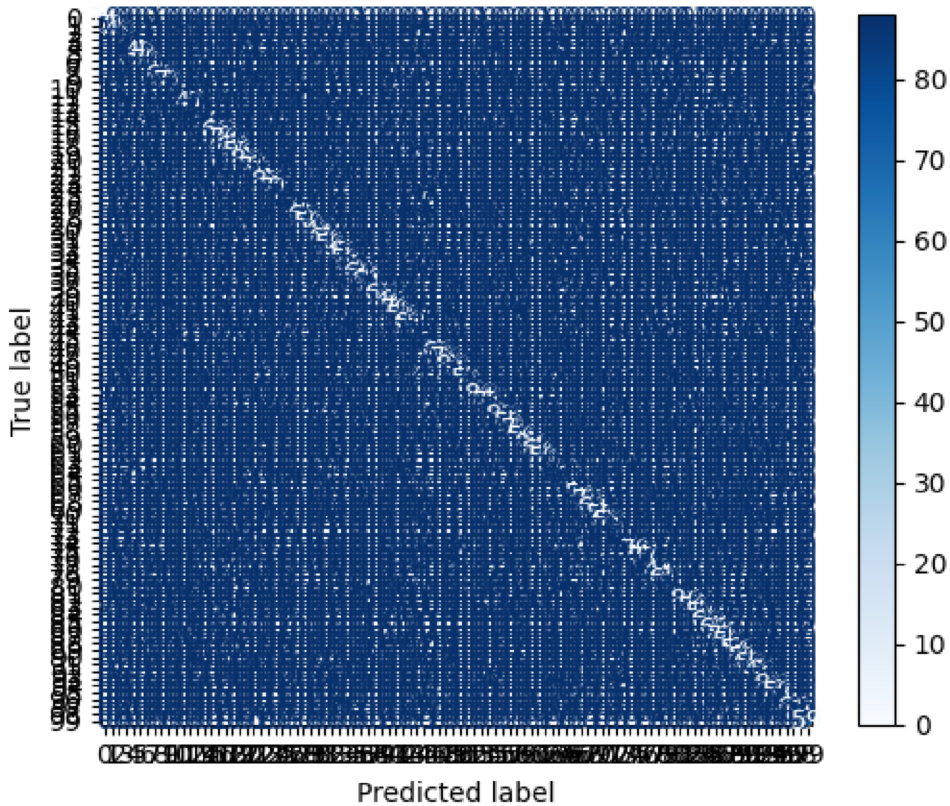
```
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and Validation Loss')
plt.ylim([0, 2])  # Adjust the y-axis limits as needed
plt.legend(loc='upper right')

plt.show()
```

```
313/313 - 1s - loss: 2.5602 - accuracy: 0.5455 - 1s/epoch - 5ms/step

Test accuracy: 0.5454999804496765
```



[24]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np
import matplotlib.pyplot as plt

# Predict classes for test images
predictions = model.predict(test_images)
predicted_classes = np.argmax(predictions, axis=1)

# Compute confusion matrix
cm = confusion_matrix(test_labels, predicted_classes)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.
 ↪unique(test_labels))
disp.plot(cmap=plt.cm.Blues)
plt.show()
```

```
313/313 [==============================] - 1s 4ms/step
```


True label
Predicted label
0
10
20
30
40
50
60
70
80