```
from tensorflow.python.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from matplotlib import pyplot as plt
from resnet import ResNet_N
from darse import Parser
from os import getpid

BATCH_SIZE = 128
EPOCHS = 200

# CIFAR_10
dataset_10 = {
    "train-01": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/data_batch_
1",
    "train-02": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/data_batch_
2",
    "train-03": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/data_batch_
3",
    "train-04": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/data_batch_
4",
    "train-05": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/data_batch_
5",
    "test": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR10_DATASET/pkl/test_batch",
}

# CIFAR_100
dataset_100 = {
    "train": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR100_DATASET/pkl/train",
    "test": "/zooper2/amaan.rahman/ECE472-DeepLearning/datasets/CIFAR100_DATASET/pkl/test",
}

# https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar
-10-photo-classification/
def plot_diagnostics(history):
    # plot loss
    plt.subplot(211)
    plt.title("Cross Entropy Loss")
    plt.plot(history.history["loss"], color="blue", label="train")
    plt.plot(history.history["val_loss"], color="orange", label="validation")
    # plot accuracy
    plt.subplot(212)
    plt.title("Classification Accuracy")
    plt.plot(history.history["accuracy"], color="blue", label="train")
    plt.plot(history.history["val_accuracy"], color="orange", label="validation")
    plt.legend()
    plt.tight_layout()
    # save plot to file
    plt.savefig(
        "/zooper2/amaan.rahman/ECE472-DeepLearning/assign4/E1_cifar100_plot_"
        + str(getpid())
        + ".png"
    )
    plt.close()


def gen_data(cifar_type):
    dataset = dataset_100 if cifar_type == "CIFAR_100" else dataset_10

    # parse
    cifar_parser = Parser(dataset, cifar_type)
    train, test = cifar_parser.parse()
```

```
    train_data, train_labels = train
    test_data, test_labels = test

    # convert labels to one-hot format
    train_labels = to_categorical(train_labels)
    test_labels = to_categorical(test_labels)

    return train_data, train_labels, test_data, test_labels


def main():
    # dataset parse
    train_data, train_labels, test_data, test_labels = gen_data("CIFAR_100")
    STEPS = 0.8 * train_data.shape[0] // BATCH_SIZE

    # model init
    model = ResNet_N(
        in_shape=(test_data.shape[1], test_data.shape[2], 3),
        layers=[2, 2, 2, 2],
        classes=100,
    )
    model.summary()

    # model compile
    # https://towardsdatascience.com/super-convergence-with-cyclical-learning-ra
tes-in-tensorflow-c1932b858252
    # https://arxiv.org/pdf/1506.01186.pdf
    model.compile(
        optimizer=Adam(),
        loss="categorical_crossentropy",
        metrics=["top_k_categorical_accuracy", "accuracy"],
    )

    # fit
    callback = ReduceLROnPlateau(monitor="val_loss", min_lr=1e-4, verbose=1)
    history = model.fit(
        x=train_data,
        y=train_labels,
        batch_size=BATCH_SIZE,
        epochs=EPOCHS,
        steps_per_epoch=STEPS,
        callbacks=[callback],
        validation_split=0.2,
    )

    # evaluate
    model.evaluate(x=test_data, y=test_labels)
    plot_diagnostics(history)


if __name__ == "__main__":
    main()
```