SOURCE CODE

TRAINING

```
import matplotlib
matplotlib.use("Agg")
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array
from keras.utils import to_categorical
from pyimagesearch.lenet import LeNet
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import cv2
import os
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required=True,
help="path to input dataset")
ap.add_argument("-m", "--model", required=True,
help="path to output model")
ap.add_argument("-p", "--plot", type=str, default="plot.png",
help="path to output loss/accuracy plot")
args = vars(ap.parse_args())
EPOCHS = 25
INIT_LR = 1e-3
BS = 32
print("[INFO] loading images...")
data = []
labels = []
```

```
imagePaths = sorted(list(paths.list_images(args["dataset"])))
random.seed(42)
random.shuffle(imagePaths)
for imagePath in imagePaths:
image = cv2.imread(imagePath)
image = cv2.resize(image, (28, 28))
image = img_to_array(image)
data.append(image)
label = imagePath.split(os.path.sep)[-2]
label = 1 if label == "Melonoma" else 0
labels.append(label)
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
(trainX, testX, trainY, testY) = train_test_split(data,
labels, test_size=0.25, random_state=42)
trainY = to_categorical(trainY, num_classes=2)
testY = to_categorical(testY, num_classes=2)
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True, fill_mode="nearest")
print("[INFO] compiling model...")
model = LeNet.build(width=28, height=28, depth=3, classes=2)
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
metrics=["accuracy"])
print("[INFO] training network...")
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
validation_data=(testX, testY), steps_per_epoch=len(trainX) // BS,
epochs=EPOCHS, verbose=1)
print("[INFO] serializing network...")
model.save(args["model"])
```

```
plt.style.use("ggplot")
plt.figure()
N = EPOCHS
plt.plot(np.arange(0, N), H.history["loss"], label="train_loss")
plt.plot(np.arange(0, N), H.history["val_loss"], label="val_loss")
plt.plot(np.arange(0, N), H.history["acc"], label="train_acc")
plt.plot(np.arange(0, N), H.history["val_acc"], label="val_acc")
plt.title("Training Loss and Accuracy on Melonoma /Not_Melonoma")
plt.xlabel("Epoch #")
plt.ylabel("Loss/Accuracy")
plt.legend(loc="lower left")
plt.savefig(args["plot"])
```

TESTING

```
from keras.preprocessing.image import img_to_array
from keras.models import load_model
import numpy as np
import argparse
import imutils
import cv2
ap = argparse.ArgumentParser()
ap.add_argument("-m", "--model", required=True,
help="path to trained model model")
ap.add_argument("-i", "--image", required=True,
help="path to input image")
args = vars(ap.parse_args())
image = cv2.imread(args["image"])
orig = image.copy()
image = cv2.resize(image, (28, 28))
image = image.astype("float") / 255.0
```

```
image = img_to_array(image)
image = np.expand_dims(image, axis=0)
print("[INFO] loading network...")
model = load_model(args["model"])
(Not_Melonoma, Melonoma) = model.predict(image)[0]
label = "Melonoma" if Melonoma > Not_Melonoma else "Not_Melonoma"
proba = Melonoma if Melonoma > Not_Melonoma else Not_Melonoma
label = "{}: {:.2f}%".format(label, proba * 100)
output = imutils.resize(orig, width=800)
cv2.putText(output, label, (10, 25), cv2.FONT_HERSHEY_SIMPLEX,
0.7, (0, 255, 0), 2)
cv2.imshow("Output", output)
cv2.waitKey(0)
```