به نام خدا

معصومه پاسبانی 99243022

تمرین سری دوم مبانی بینایی

سوال اول)

ابتدا لیست کلاس‌ها از روی نام پوشه‌ها استخراج شده و به هر کلاس یک برچسب عددی اختصاص داده می‌شود. تصاویر آموزشی و آزمایشی به همراه برچسب‌های مربوطه بارگذاری می‌شوند. تصاویر به صورت آرایه‌های عددی چندبعدی خوانده می‌شوند و برچسب‌ها نیز به آرایه‌ای تبدیل می‌شوند. سپس، شکل داده‌های آموزشی و آزمایشی بررسی می‌شود تا اطمینان حاصل شود که به درستی بارگذاری شده‌اند.

داده ها با استفاده از کتابخانه‌های `os` و `shutil`از حالت فشرده خارج شدند. تصاویر در دو مجموعه آموزشی و آزمایشی دسته‌بندی شده‌اند، به طوری که هر کلاس (دسته) از تصاویر در یک پوشه جداگانه قرار دارد.

```
] x_train = []
  y_train = []

  x_test = []
  y_test = []

  for _class, lable in zip(classes, class_lables):
    path_images = os.listdir("shoes/train/"+_class)
    path_images = ["shoes/train/" + _class + "/" + img_path for img_path in path_images]
    x_train.extend([cv.cvtColor(cv.imread(img), cv.COLOR_BGR2RGB) for img in path_images])
    y_train.extend(np.zeros(len(path_images)) + lable)

    path_images = os.listdir("shoes/test/"+_class)
    path_images = ["shoes/test/" + _class + "/" + img_path for img_path in path_images]
    x_test.extend([cv.cvtColor(cv.imread(img), cv.COLOR_BGR2RGB) for img in path_images])
    y_test.extend(np.zeros(len(path_images)) + lable)

  x_train = np.array(x_train)
  y_train = np.array(y_train)
  x_test = np.array(x_test)
  y_test = np.array(y_test)

  x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

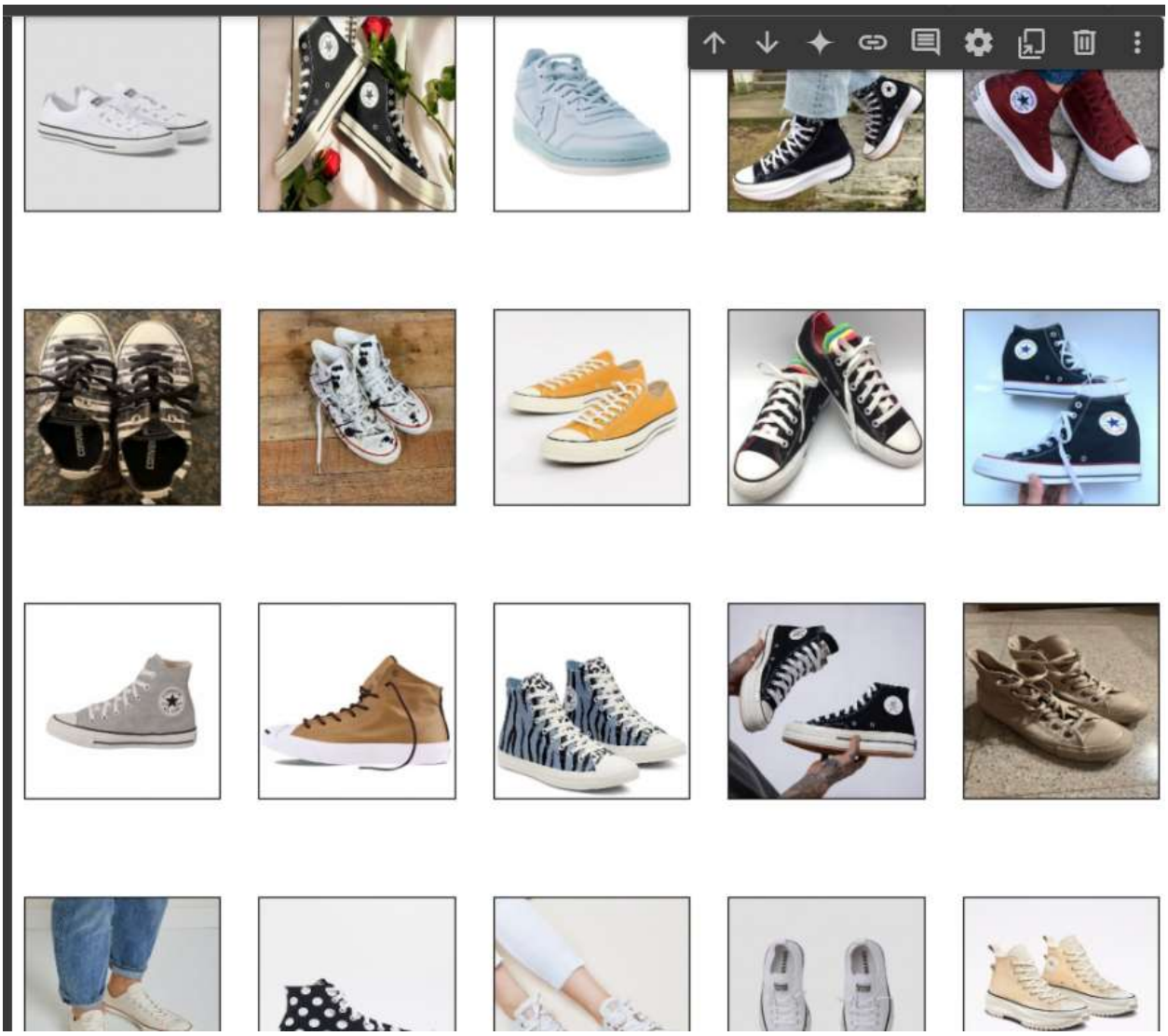
برای مدل‌سازی بهتر، از ساختاری استفاده شد که هر کلاس به یک برچسب عددی یکتا اختصاص داده شود. به همین منظور، نام پوشه‌ها به عنوان برچسب دسته‌ها استخراج شد و یک دیکشنری برای نگهداری این نگاشت (نام کلاس‌ها به اعداد) ساخته شد. این کار باعث می‌شود که داده‌ها به صورت یکنواخت برای مدل آماده شوند.

.تمامی تصاویر به آرایه‌های چندبعدی تبدیل شدند تا به‌راحتی به مدل‌های یادگیری عمیق ورودی داده شوند. همچنین، اندازه تصاویر به یک مقدار ثابت) ۲۲۴x۲۲۴ پیکسل (تغییر داده شد تا مدل بتواند ورودی‌های یکنواخت دریافت کند. علاوه بر این، داده‌ها به آرایه‌های ویژگی‌ها (X) و برچسب‌ها (y) تقسیم‌بندی شدند.

برای آموزش و ارزیابی مدل، داده‌ها به دو بخش training set و testing setتقسیم شدند. از روش‌های استاندارد تقسیم‌بندی، مانند train_test_split از کتابخانه Scikit-learn استفاده شد. در این تقسیم‌بندی، به گونه‌ای عمل شد که توزیع کلاس‌ها در هر دو مجموعه متعادل باشد.

از کتابخانه Matplotlib برای نمایش تصادفی چند تصویر از مجموعه آموزشی استفاده شد. این کار به درک بصری بهتر از داده‌ها و شناسایی مشکلات احتمالی مانند نویز یا کیفیت پایین تصاویر کمک کرد. همچنین، توزیع تعداد نمونه‌های هر کلاس با استفاده از نمودارهای میله‌ای تجسم شد تا از تعادل نسبی داده‌ها اطمینان حاصل شود.

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(12, 12))

for i in range(4):
  plt.subplot(4, 5, i*5 +1, xticks=[], yticks=[])
  plt.imshow(x_train[i*5+0], cmap=plt.cm.binary_r)

  plt.subplot(4, 5, i*5 +2, xticks=[], yticks=[])
  plt.imshow(x_train[i*5+1], cmap=plt.cm.binary_r)

  plt.subplot(4, 5, i*5 +3, xticks=[], yticks=[])
  plt.imshow(x_train[i*5+2], cmap=plt.cm.binary_r)

  plt.subplot(4, 5, i*5 +4, xticks=[], yticks=[])
  plt.imshow(x_train[i*5+3], cmap=plt.cm.binary_r)

  plt.subplot(4, 5, i*5 +5, xticks=[], yticks=[])
  plt.imshow(x_train[i*5+4], cmap=plt.cm.binary_r)
```



برای افزایش تنوع داده‌ها و بهبود عملکرد مدل، از روش‌های تقویت داده استفاده شد. این مرحله با استفاده از ابزار ImageDataGeneratorاز کتابخانه Keras پیاده‌سازی شد. تقویت داده‌ها شامل موارد زیر بود:

چرخش تصاویر: تصاویر در زوایای مختلف چرخانده شدند تا مدل به تغییرات زاویه‌ای مقاوم شود.

اعمال نویز: نویز به تصاویر اضافه شد تا مدل در برابر نویز مقاوم‌تر شود.

جابه‌جایی افقی و عمودی: تصاویر به صورت افقی و عمودی تغییر مکان داده شدند.

تغییر روشنایی و کنتراست: با تغییر پارامترهای روشنایی و کنتراست، تنوع داده‌ها افزایش یافت.

```python
def plot_loss(history):
  train_loss = history.history['loss']
  valid_loss = history.history['val_loss']
  epochs = range(1, len(train_loss) + 1)

  plt.plot(epochs, train_loss, 'g-', label='Training loss')
  plt.plot(epochs, valid_loss, 'b-', label='Validation loss')
  plt.title('Training and Validation Loss'), plt.xlabel('Epochs'), plt.ylabel('Loss'), plt.legend(), plt.grid(True), plt.show()
  plt.show()


from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix


def calculate_metrics(true_lables, predicted_lables):
  return {
    'Accuracy': accuracy_score(true_lables, predicted_lables),
    'Precision': precision_score(true_lables, predicted_lables, average='weighted'),
    'Recall': recall_score(true_lables, predicted_lables, average='weighted'),
    'F1': f1_score(true_lables, predicted_lables, average='weighted'),
  }


def plot_cm(model, x, y):
  y_pred = []
  y_pred = model.predict(x)
  y_pred = [y.argmax() for y in y_pred]
  print(calculate_metrics(y, y_pred))

  cm = confusion_matrix(y, y_pred)

  cm_normalized = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
```

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator


seed = 13
np.random.seed(seed)
tf.random.set_seed(seed)


def add_noise(image):
    noise = np.random.normal(loc=0, scale=0.1, size=image.shape)
    return np.clip(image + noise, 0, 1)


data_generator = ImageDataGenerator(
  # preprocessing_function=add_noise,
  rotation_range=5,
  width_shift_range=0.02,
  height_shift_range=0.02,
  shear_range=0.02,
  zoom_range=0.02,
  horizontal_flip=True,
  fill_mode='constant'
)

generator = data_generator.flow(x_train, y_train, batch_size=4)
```

تقویت داده‌ها باعث شد که مدل به‌جای حفظ ویژگی‌های خاص یک نمونه، ویژگی‌های کلی‌تر دسته‌ها را یاد بگیرد.

```
x = Conv2D(filters=32, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(inputs)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)

x = Conv2D(filters=32, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv2D(filters=32, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv2D(filters=32, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv2D(filters=64, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
x = MaxPooling2D(pool_size=(2, 2))(x)

x = Conv2D(filters=64, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv2D(filters=64, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)

x = Conv2D(filters=64, kernel_size=3, kernel_regularizer=l2(1e-5), bias_regularizer=l2(1e-5), strides=1)(x)
x = BatchNormalization()(x)
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 240, 240, 3)	0
conv2d (Conv2D)	(None, 238, 238, 32)	896
batch_normalization (BatchNormalization)	(None, 238, 238, 32)	128
activation (Activation)	(None, 238, 238, 32)	0
max_pooling2d (MaxPooling2D)	(None, 119, 119, 32)	0
conv2d_1 (Conv2D)	(None, 117, 117, 32)	9,248
batch_normalization_1 (BatchNormalization)	(None, 117, 117, 32)	128
activation_1 (Activation)	(None, 117, 117, 32)	0
conv2d_2 (Conv2D)	(None, 115, 115, 32)	9,248
batch_normalization_2 (BatchNormalization)	(None, 115, 115, 32)	128
activation_2 (Activation)	(None, 115, 115, 32)	0
conv2d_3 (Conv2D)	(None, 113, 113, 32)	9,248
batch_normalization_3 (BatchNormalization)	(None, 113, 113, 32)	128
activation_3 (Activation)	(None, 113, 113, 32)	0
conv2d_4 (Conv2D)	(None, 111, 111, 64)	18,496

یک مدل پایه با استفاده از شبکه عصبی کانولوشن (CNN) طراحی شد. این مدل شامل چندین لایه کانولوشن و ماکس‌پولینگ بود که به‌صورت متوالی برای استخراج ویژگی‌ها و کاهش ابعاد داده‌ها استفاده شدند.

لایه‌های کانولوشن: این لایه‌ها با استفاده از فیلترهای کانولوشن، ویژگی‌های محلی تصاویر را استخراج می‌کنند. از تابع فعال‌سازی ReLU برای افزایش غیرخطی بودن استفاده شد.

لایه ماکس‌پولینگ: برای کاهش ابعاد و پیچیدگی محاسبات، از این لایه‌ها استفاده شد. این لایه‌ها اطلاعات مهم تصاویر را حفظ کرده و جزئیات غیرضروری را حذف می‌کنند.

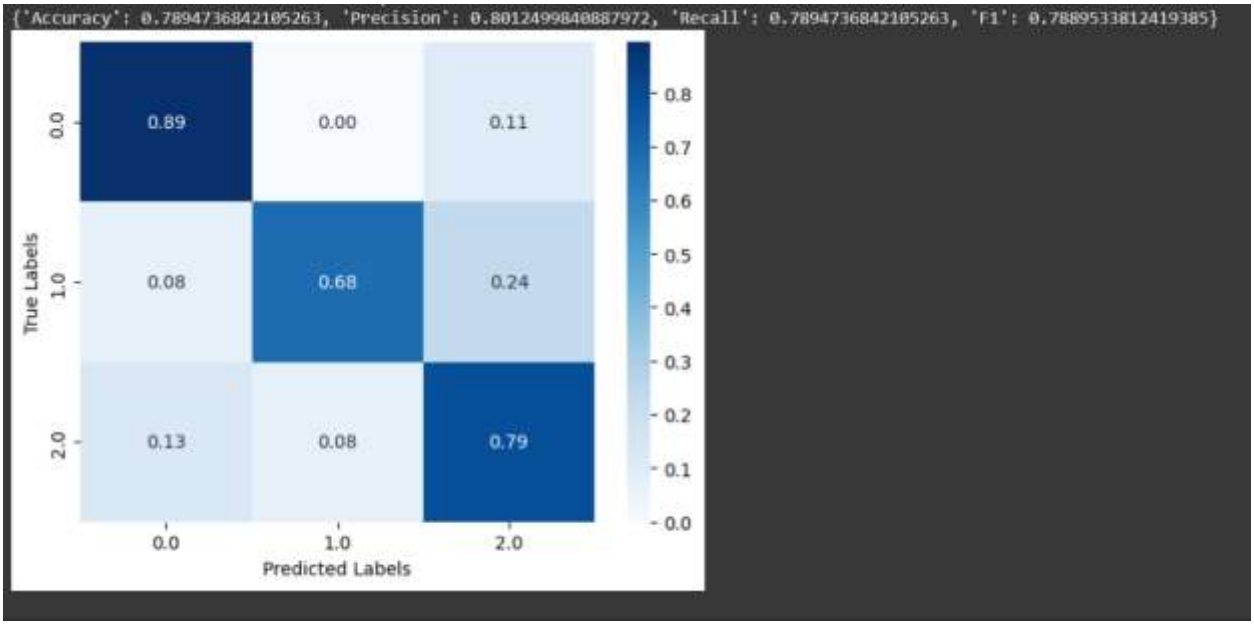
لایه‌های Dense: در انتها، داده‌ها به لایه‌های کاملاً متصل (Dense) وارد شدند. این لایه‌ها خروجی مدل را به تعداد کلاس‌های دسته‌بندی تنظیم کردند.

از الگوریتم Adam برای بهینه‌سازی وزن‌ها استفاده شد، زیرا سرعت همگرایی بالایی دارد.تابع Loss به صورت categorical_crossentropyتعریف شد که برای مسائل دسته‌بندی چندکلاسه مناسب است.معیار ارزیابی دقت Accuracyنیز در هنگام آموزش مشخص شد.

```
model3.compile(optimizer=Adam(learning_rate = 0.0001), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

earlyStop = EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto')

check_point_name = 'best_weight.keras'
check_point = ModelCheckpoint(filepath = check_point_name, monitor='val_loss', verbose=1, save_best_only=True, mode='auto')

history3 = model3.fit(
  generator,
  epochs=15,
  verbose=1,
  validation_data=(x_test, y_test),
  callbacks=[earlyStop, check_point],
  shuffle=True
)
```

سپس تعداد اپوک‌ها (epochs) و اندازه دسته (batch size) تنظیم شدند.برای جلوگیری از بیش‌برازش، از `EarlyStopping`استفاده شد. این ابزار فرآیند آموزش را در صورت عدم بهبود عملکرد مدل روی داده‌های اعتبارسنجی متوقف می‌کند.

وزن‌های مدل در هر اپوک ذخیره شدند تا بهترین نسخه مدل نگهداری شود.



پس از آموزش، مدل بر روی داده‌های آزمایشی ارزیابی شد. معیارهای زیر برای ارزیابی مدل استفاده شدند و accuracy ،

precision وrecall و F1 score محاسبه شدند.
برای بررسی دقیق‌تر عملکرد مدل، ماتریس سردرگمی (Confusion Matrix) ترسیم شد. این ماتریس نشان می‌دهد که مدل در پیش‌بینی هر دسته تا چه حد موفق عمل کرده است.

در نهایت برای بهبود عملکرد، از مدل ResNet استفاده شد .ResNet شامل بلوک‌های باقیمانده (Residual Blocks) است که به یادگیری در لایه‌های عمیق کمک می‌کند. این معماری با استفاده از توابع `Conv2D`، `BatchNormalization`و `Add`در Keras طراحی شد.

مزایای:ResNet

کاهش مشکل ناپدید شدن گرادیان.(Vanishing Gradient)

توانایی یادگیری ویژگی‌های پیچیده‌تر.

عملکرد بهتر در مقایسه با مدل‌های ساده‌تر.

و برای بهبود بیشتر، نرخ یادگیری در بهینه‌سازها تغییر داده شد، تعداد لایه‌ها افزایش یا کاهش یافت و از تکنیک‌های مختلف Dropoutبرای جلوگیری از بیش‌برازش استفاده شد.

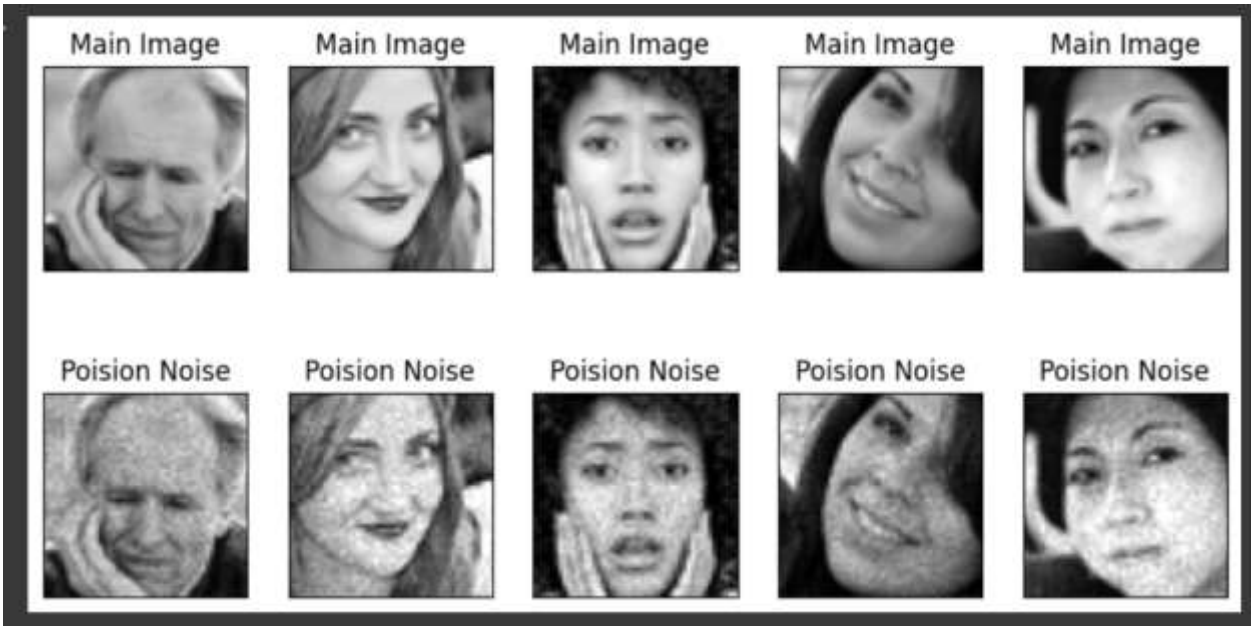
سوال دوم)

ابتدا، مجموعه داده‌ای از تصاویر چهره استفاده شد که شامل تصاویر ۴۸×۴۸ پیکسلی است. این تصاویر به سه بخش تقسیم شدند: training, validation, test. تصاویر در مجموعه داده به صورت مقادیر عددی در قالب متن ذخیره شده‌اند. این مقادیر با استفاده از تابع `str2img`به آرایه‌های دوبعدی تبدیل شده و به تصاویر ۴۸×۴۸ پیکسل شکل‌دهی می‌شوند.

در مرحله بعد، به تصاویر نویز پواسون اضافه شد تا مدل بتواند حذف نویز را بیاموزد. همچنین، تمامی تصاویر به صورت سیاه و سفید و مقادیر آن‌ها نرمال‌سازی شدند تا یادگیری بهتر انجام شود.

```python
import cv2 as cv
mean=0
sigma=10

def add_poisson_noise(img):
    image = img.copy()
    noisy_image = np.random.poisson(image).astype('float32')
    noisy_image = np.clip(noisy_image, 0, 255)
    return noisy_image

x_train = np.array([add_poisson_noise(img) for img in y_train])
x_valid = np.array([add_poisson_noise(img) for img in y_valid])
x_test = np.array([add_poisson_noise(img) for img in y_test])
```

```python
import matplotlib.pyplot as plt

np.random.seed(48)

indices = np.random.randint(0, y_test.shape[0], size=5)

fig = plt.figure(figsize=(20, 5))

for i, index in enumerate(indices):
  plt.subplot(2, 10, i + 1, xticks=[], yticks=[])
  plt.imshow(y_test[index], cmap=plt.cm.binary_r), plt.title('Main Image')
for i, index in enumerate(indices):
  plt.subplot(2, 10, i + 11, xticks=[], yticks=[])
  plt.imshow(x_test[index], cmap=plt.cm.binary_r), plt.title('Poision Noise')

plt.show()
```



برای افزایش کارایی و پایداری مدل، مقادیر پیکسل‌های تصاویر اصلی و نویزی به محدوده [۰, ۱] نرمال‌سازی شدند.

برای طراحی مدل، از ساختار U-Net استفاده شد. این مدل دو بخش اصلی دارد:

بخش فشرده‌سازی :(Encoder) تصاویر ورودی را به ویژگی‌های سطح پایین تبدیل می‌کند. این بخش شامل لایه‌های کانولوشن و MaxPooling است که ابعاد تصاویر را کاهش داده و ویژگی‌های مهم را استخراج می‌کنند.

بخش بازسازی :(Decoder) ویژگی‌های استخراج‌شده را به تصویر اصلی بازسازی‌شده تبدیل می‌کند.از لایه‌های UpSamplingو Concatenate برای بازسازی تصویر استفاده می‌شود.

```python
inputs = tf.keras.Input(shape=(48, 48, 1))

# Encoder
encoder_block1_conv1 = layers.Conv2D(32, 1, activation='relu', padding='same')(inputs)
encoder_block1_conv2 = layers.Conv2D(32, 1, activation='relu', padding='same')(encoder_block1_conv1)
encoder_block1_pool = layers.MaxPooling2D(pool_size=(2, 2))(encoder_block1_conv2)

encoder_block2_conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(encoder_block1_pool)
encoder_block2_conv2 = layers.Conv2D(64, 3, activation='relu', padding='same')(encoder_block2_conv1)
encoder_block2_pool = layers.MaxPooling2D(pool_size=(2, 2))(encoder_block2_conv2)

encoder_block3_conv1 = layers.Conv2D(128, 3, activation='relu', padding='same')(encoder_block2_pool)
encoder_block3_conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(encoder_block3_conv1)
encoder_block3_pool = layers.MaxPooling2D(pool_size=(2, 2))(encoder_block3_conv2)

# Bottleneck
bottleneck_conv1 = layers.Conv2D(256, 3, activation='relu', padding='same')(encoder_block3_pool)
bottleneck_conv2 = layers.Conv2D(256, 3, activation='relu', padding='same')(bottleneck_conv1)

# Decoder
decoder_block1_upsample = layers.UpSampling2D(size=(2, 2))(bottleneck_conv2)
decoder_block1_concat = layers.Concatenate()([decoder_block1_upsample, encoder_block3_conv2])
decoder_block1_conv1 = layers.Conv2D(128, 3, activation='relu', padding='same')(decoder_block1_concat)
decoder_block1_conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(decoder_block1_conv1)

decoder_block2_upsample = layers.UpSampling2D(size=(2, 2))(decoder_block1_conv2)
decoder_block2_concat = layers.Concatenate()([decoder_block2_upsample, encoder_block2_conv2])
decoder_block2_conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(decoder_block2_concat)
decoder_block2_conv2 = layers.Conv2D(64, 3, activation='relu', padding='same')(decoder_block2_conv1)

decoder_block3_upsample = layers.UpSampling2D(size=(2, 2))(decoder_block2_conv2)
```

```
decoder_block2_conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(dec
decoder_block2_conv2 = layers.Conv2D(64, 3, activation='relu', padding='same')(dec

decoder_block3_upsample = layers.UpSampling2D(size=(2, 2))(decoder_block2_conv2)
decoder_block3_concat = layers.Concatenate()([decoder_block3_upsample, encoder_block1_conv2])
decoder_block3_conv1 = layers.Conv2D(32, 3, activation='relu', padding='same')(decoder_block3_concat)
decoder_block3_conv2 = layers.Conv2D(32, 3, activation='relu', padding='same')(decoder_block3_conv1)

# Output layer
outputs = layers.Conv2D(1, 1, activation=None)(decoder_block3_conv2)

model = models.Model(inputs=inputs, outputs=outputs)

model.summary()
```

Model: "functional_3"

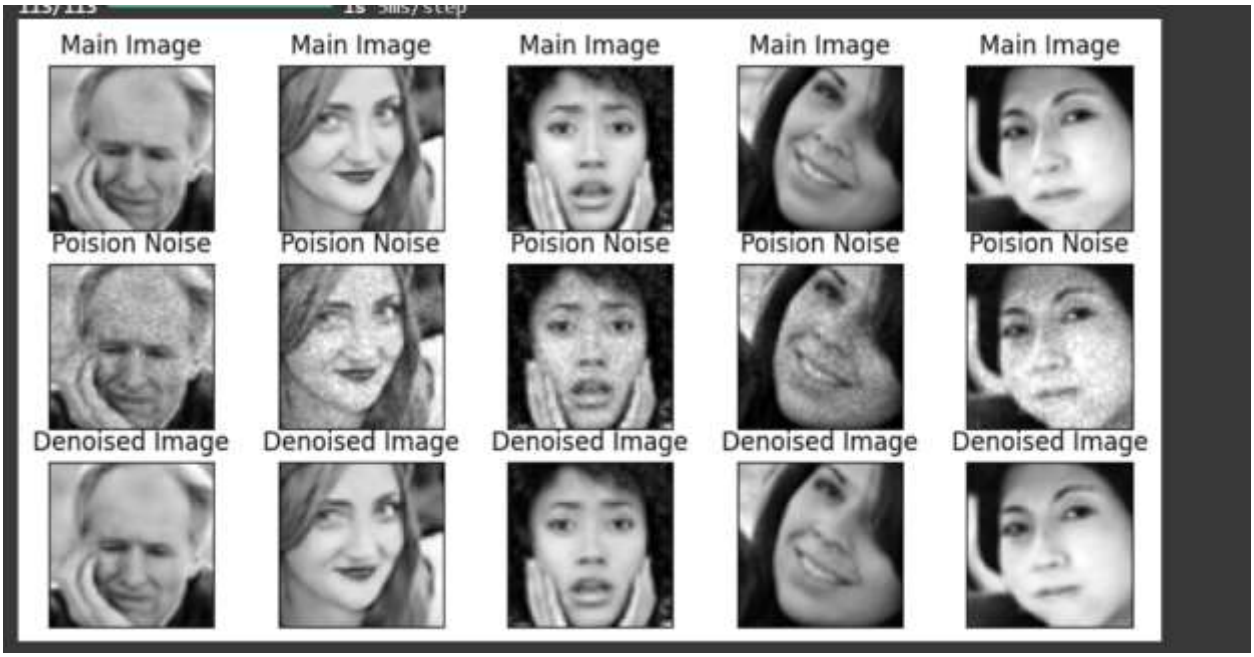
Layer (type)	Output Shape	Param #	Connected to
input_layer_6 (InputLayer)	(None, 48, 48, 1)	0	-
conv2d_75 (Conv2D)	(None, 48, 48, 32)	64	input_layer_6[0][0]
conv2d_76 (Conv2D)	(None, 48, 48, 32)	1,056	conv2d_75[0][0]
max_pooling2d_18 (MaxPooling2D)	(None, 24, 24, 32)	0	conv2d_76[0][0]
conv2d_77 (Conv2D)	(None, 24, 24, 64)	18,496	max_pooling2d_18[0][0]
conv2d_78 (Conv2D)	(None, 24, 24, 64)	36,928	conv2d_77[0][0]
max_pooling2d_19 (MaxPooling2D)	(None, 12, 12, 64)	0	conv2d_78[0][0]

مدل با استفاده از معیار خطای میانگین مربعات (MSE) آموزش داده شد تا اختلاف میان تصاویر ورودی و بازسازی‌شده را به حداقل برساند. در فرآیند آموزش، از داده‌های آموزشی برای یادگیری و از داده‌های اعتبارسنجی برای ارزیابی عملکرد مدل استفاده شد. همچنین، مکانیزم‌هایی مانند (EarlyStopping)برای جلوگیری از بیش‌برازش به کار رفت.

```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam

early_stop = EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto')
checkpoint_filename = 'best_denoising_weights.keras'
model_checkpoint = ModelCheckpoint(filepath=checkpoint_filename, monitor='val_loss', verbose=1, save_best_only=True
optimizer = Adam(learning_rate=0.0001)

model.compile(optimizer=optimizer, loss='mean_squared_error', metrics=['mean_squared_error'])

history = model.fit(
  x_train,
  y_train,
  epochs=20,
  batch_size=32,
  validation_data=(x_valid, y_valid),
  callbacks=[early_stop, model_checkpoint],
  verbose=1
)
```

در پایان، عملکرد مدل روی داده‌های آزمون ارزیابی شد. تصاویر بازسازی‌شده نشان دادند که مدل توانسته بخش زیادی از نویز را حذف کند و تصاویر تقریباً شبیه به تصاویر اصلی شوند. البته، ممکن است برخی جزئیات دقیق در تصاویر بازسازی‌شده از دست برود یا بخشی از نویز باقی بماند.

```
model_output = model.predict(x_test)

np.random.seed(48)

indices = np.random.randint(0, y_test.shape[0], size=5)

fig = plt.figure(figsize=(20, 5))

for i, index in enumerate(indices):
  plt.subplot(3, 10, i + 1, xticks=[], yticks=[])
  plt.imshow(y_test[index], cmap=plt.cm.binary_r), plt.title('Main Image')
for i, index in enumerate(indices):
  plt.subplot(3, 10, i + 11, xticks=[], yticks=[])
  plt.imshow(x_test[index], cmap=plt.cm.binary_r), plt.title('Poision Noise')
for i, index in enumerate(indices):
  plt.subplot(3, 10, i + 21, xticks=[], yticks=[])
  plt.imshow(model_output[index], cmap=plt.cm.binary_r), plt.title('Denoised Image')
```

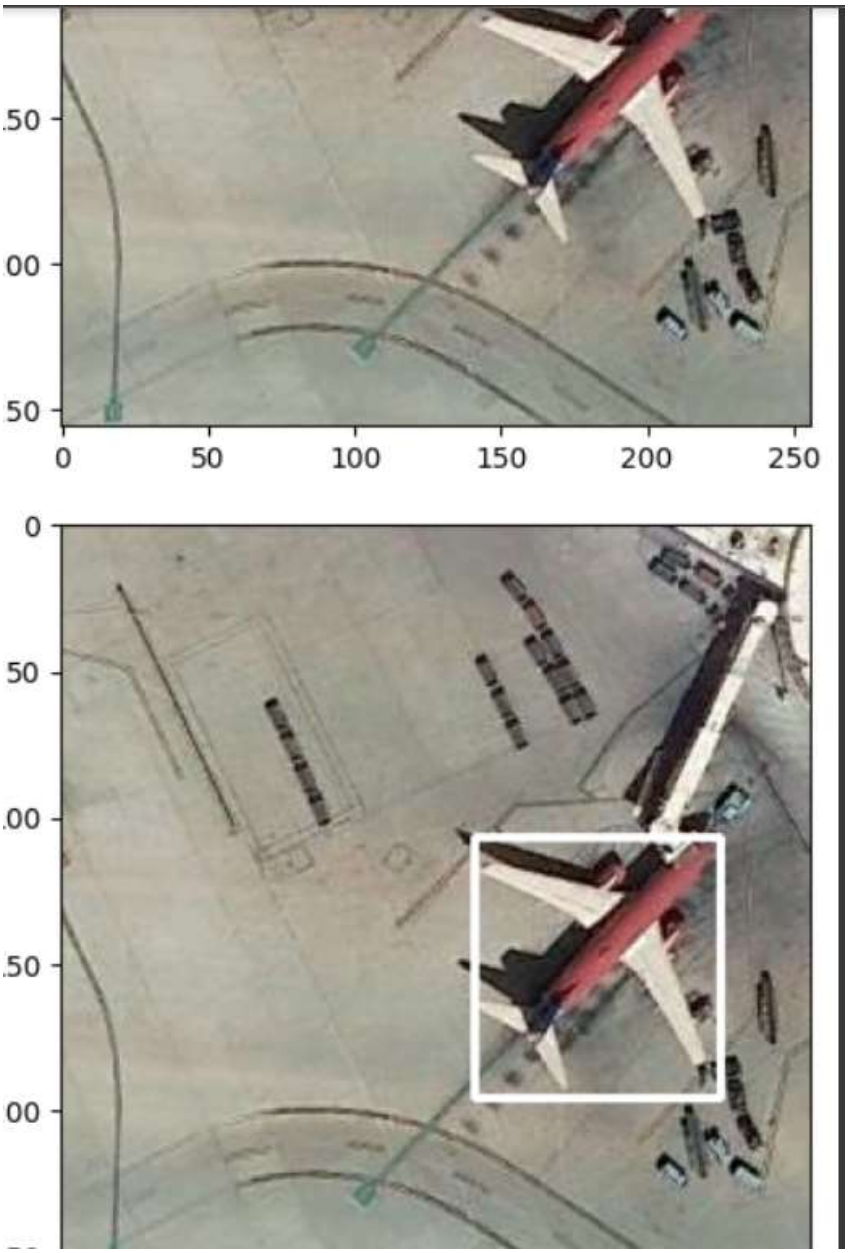


نتیجه کلی نشان می‌دهد که این مدل U-Net به خوبی توانسته وظیفه حذف نویز را انجام دهد و می‌تواند در کاربردهایی مثل پردازش تصاویر پزشکی یا تصویربرداری صنعتی مفید باشد. با این حال، برای بهبود بیشتر می‌توان از تکنیک‌هایی مثل غنی‌سازی داده‌ها یا استفاده از معیارهای پیشرفته‌تر برای ارزیابی کیفیت تصاویر استفاده کرد.

سوال سوم)

در ابتدا داده‌ها از Google Drive بارگذاری شده‌اند.کتابخانه‌های OpenCV برای پردازش تصاویر و تغییر اندازه آن‌ها استفاده شده‌اند و مختصات جعبه‌های محدودکننده از فایل‌های حاشیه‌نویسی خوانده و روی تصاویر نمایش داده شده‌اند.

```python
def data_viz(rand_num):
    img_name = 'airplane_'+str(rand_num) +'.jpg'
    img = cv2.imread(os.path.join(img_path,img_name))
    data_frame = pd.read_csv(os.path.join(annotations_path,img_name.replace('.jpg','.csv')))
    plt.imshow(img)

    for r in data_frame.iterrows():
        p1 = r[1][0].split(' ')[0]
        q1 = r[1][0].split(' ')[1]
        p2 = r[1][0].split(' ')[2]
        q2 = r[1][0].split(' ')[3]

        p1 = int(p1)
        q1 = int(q1)
        p2 = int(p2)
        q2 = int(q2)

        img_bb = cv2.rectangle(img,(p1,q1),(p2,q2),(255,255,255),2)
        plt.figure()
        plt.imshow(img_bb)

    return img_bb
```

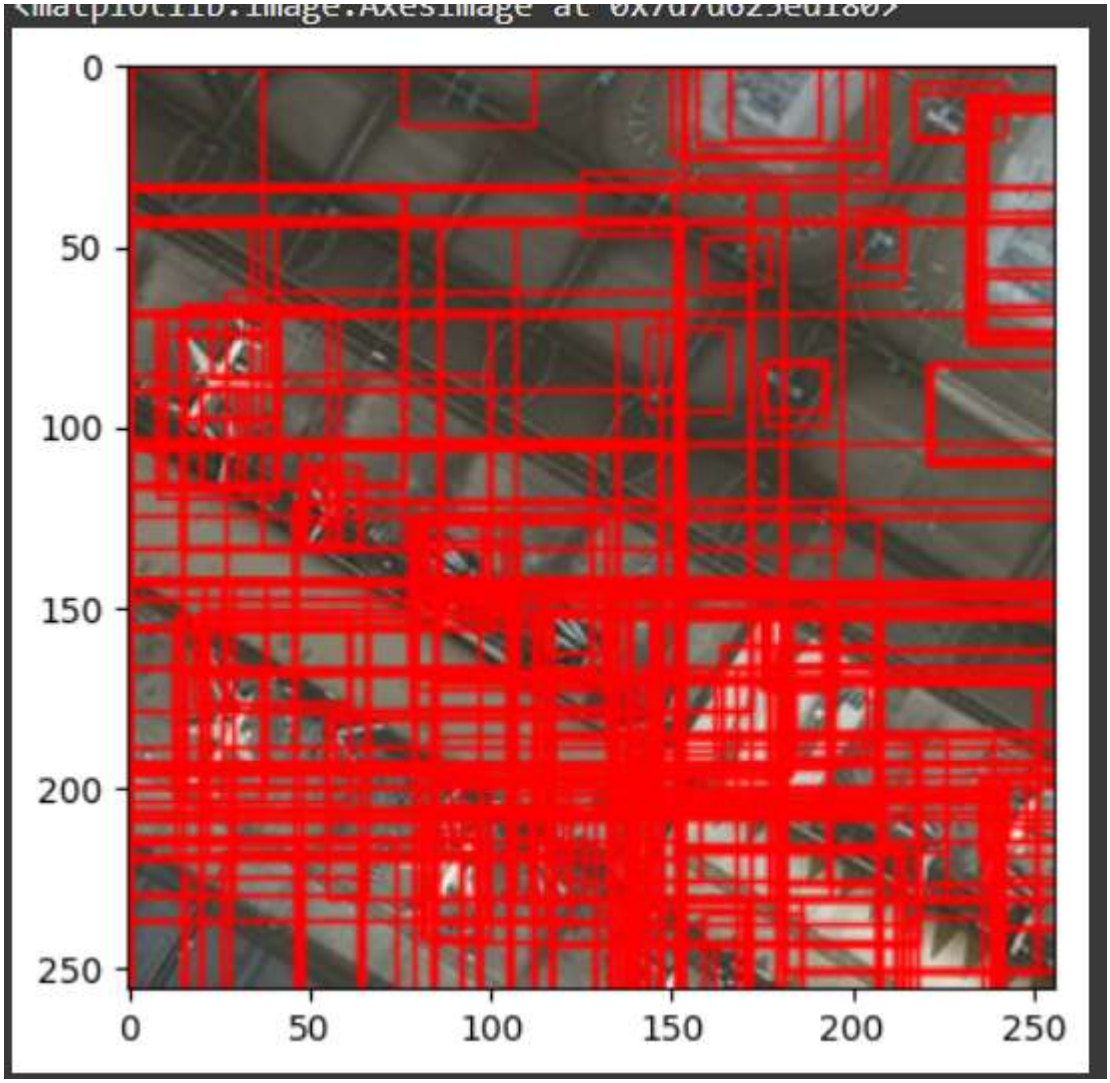


تابع برای نمایش تصاویر با جعبه‌های محدودکننده توسعه داده شده است. این کار به‌منظور اطمینان از صحت داده‌های حاشیه‌نویسی و نمایش نواحی واقعی انجام شده است. این جعبه‌ها نشان‌دهنده مکان دقیق هواپیماها در تصاویر هستند.

از الگوریتم Selective Search برای شناسایی نواحی پیشنهادی استفاده شده است. این الگوریتم تصویر را به قسمت‌های کوچک تقسیم می‌کند.بخش‌های مشابه براساس ویژگی‌هایی مانند رنگ، بافت و اندازه با هم ترکیب می‌شوند.خروجی این الگوریتم، نواحی مستطیلی پیشنهادی Region Proposalsاست که احتمال وجود هواپیما در آن‌ها بالا است.

```
cv2.setUseOptimized(True);
sel_search_seg = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()
img1 = cv2.imread(os.path.join(img_path,'42845.jpg'))
sel_search_seg.setBaseImage(img1)
sel_search_seg.switchToSelectiveSearchFast()
rects = sel_search_seg.process()
img_output= img1.copy()
img_out = img1.copy()
```

```
for i,rect in enumerate((rects)):
    x,y,w,h = rect
    print(x,y,w,h)

    cv2.rectangle(img_out,(x,y),(x+w,y+h),(255,0,0),1,cv2.LINE_AA)

plt.imshow(img_out)
```



```python
def IOU(bb1,bb2):
    assert bb1['x1'] < bb1['x2']
    assert bb1['y1'] < bb1['y2']
    assert bb2['x1'] < bb2['x2']
    assert bb2['y1'] < bb2['y2']

    x_left = max(bb1['x1'], bb2['x1'])
    x_right = min(bb1['x1'],bb2['x2'])
    y_bottom = min(bb1['y2'],bb2['y2'])
    y_top = max(bb1['y1'],bb2['y2'])

    if x_right < x_left or y_bottom < y_top:
        return 0.

    intersect_area = (x_right - x_left)*(y_bottom - y_top)
    bb1_area =  (bb1['x2']- bb1['x1']) * (bb1['y2'] - bb1['y1'])
    bb2_area =  (bb2['x2']- bb2['x1']) * (bb2['y2'] - bb2['y1'])

    IOU = intersect_area/float(bb1_area + bb2_area - intersect_area)
    assert IOU >=0.
    assert IOU <=1.

    return IOU
```

معیار **IoU (Intersection over Union)**برای ارزیابی نواحی پیشنهادی استفاده شده است؛ اگر IoU بین یک ناحیه پیشنهادی و یک جعبه محدودکننده واقعی بیش از 0.7 باشد، ناحیه به عنوان **مثبت (Positive)** در نظر گرفته می‌شود.اگر IoUکمتر از 0.3 باشد، ناحیه به عنوان **منفی (Negative)** در نظر گرفته می‌شود.این معیار برای تعیین کیفیت نواحی پیشنهادی ضروری است.

```
for e,name in enumerate(os.listdir(annotations_path)):
    if name.startswith('airplane'):
        file_name = name.split('.')[0]+'.jpg'
        print(e,file_name)
        img = cv2.imread(os.path.join(img_path,file_name))
        dataframe = pd.read_csv(os.path.join(annotations_path,name))
        getvalues = []
        for r in dataframe.iterrows():
            x1 = r[1][0].split(' ')[0]
            y1 = r[1][0].split(' ')[1]
            x2 = r[1][0].split(' ')[2]
            y2 = r[1][0].split(' ')[3]

            x1 = int(x1)
            y1 = int(y1)
            x2 = int(x2)
            y2 = int(y2)

            getvalues.append({'x1':x1,'x2':x2,'y1':y1,'y2':y2})

        sel_search_seg.setBaseImage(img)
        sel_search_seg.switchToSelectiveSearchFast()
        sel_search_seg_res = sel_search_seg.process()

        img_out = img.copy()
        count = 0
        false_count = 0
        flag = 0
        fflag = 0
        bflag = 0
```

```
for e, res in enumerate(sel_search_seg_res):
    try:

        if e < 2000 and flag == 0:
            for gval in getvalues :
                x,y,w,h = res
                iou = IOU(gval,{'x1':x1,'x2':x+w,'y1':y,'y2':y+h})

                if count < 30:
                    if iou > 70:
                        temp_img = img_out[x:x+w,y:y+h]
                        resized = cv2.resize(temp_img,(224,224),interpolation = cv2.INTER_AREA)

                        X_train.append(resized)
                        y_train.append(0)
                        false_count = false_count+1
                    else:
                        fflag = 1
                    if false_count <30:
                        if iou<0.3:
                            temp_img = img_out[x:x+w,y:y+h]
                            resized = cv2.resize(temp_img,(224,224),interpolation= cv2.INTER_AREA)
                            X_train.append(resized)
                            y_train.append(0)
                            false_count = false_count+1

                        else:
                            bflag = 1

                if fflag==1 and bflag ==1:

                    print('in')
```

نواحی مثبت و منفی به عنوان نمونه‌های جداگانه ذخیره شده‌اند.تصاویر به اندازه 224 224xتغییر اندازه داده شده‌اند (اندازه موردنیاز برای مدل‌های از پیش‌آموزش‌دیده).این مجموعه داده‌ها برای آموزش مدل‌های یادگیری عمیق استفاده شده‌اند.

مدل‌های ResNet50 و VGG16 به‌عنوان استخراج‌کننده ویژگی استفاده شده‌اند.این مدل‌ها از قبل روی مجموعه داده

ImageNetآموزش دیده‌اند و برای این پروژه تنظیم (Fine-tuned) شده‌اند.ویژگی‌های نواحی پیشنهادی توسط این مدل‌ها استخراج شده و به لایه‌های Fully Connected ارسال می‌شوند.مدل نهایی برای تشخیص وجود هواپیما در نواحی پیشنهادی آموزش داده شده است.

```
for e,i in enumerate(os.listdir(annotations_path)):
    try:
        if i.startswith("airplane"):
            filename = i.split(".")[0]+".jpg"
            print(e,filename)
            image = cv2.imread(os.path.join(img_path,filename))
            df = pd.read_csv(os.path.join(annotations_path,i))
            gtvalues=[]
            for row in df.iterrows():
                x1 = int(row[1][0].split(" ")[0])
                y1 = int(row[1][0].split(" ")[1])
                x2 = int(row[1][0].split(" ")[2])
                y2 = int(row[1][0].split(" ")[3])
                gtvalues.append({"x1":x1,"x2":x2,"y1":y1,"y2":y2})
                timage = image[x1:x2,y1:y2]
                resized = cv2.resize(timage, (224,224), interpolation = cv2.INTER_AREA)
                svm_imgs.append(resized)
                svm_labels.append([0,1])
            sel_search_seg.setBaseImage(image)
            sel_search_seg.switchToSelectiveSearchFast()
            sel_search_seg_res = sel_search_seg.process()
            imout = image.copy()
            counter = 0
            falsecounter = 0
            flag = 0
            for e,result in enumerate(sel_search_seg_res):
                if e < 2000 and flag == 0:
                    for gtval in gtvalues:
                        x,y,w,h = result
                        iou = IOU(gtval,{"x1":x,"x2":x+w,"y1":y,"y2":y+h})
                        if falsecounter <5:
                            if iou < 0.3:
```

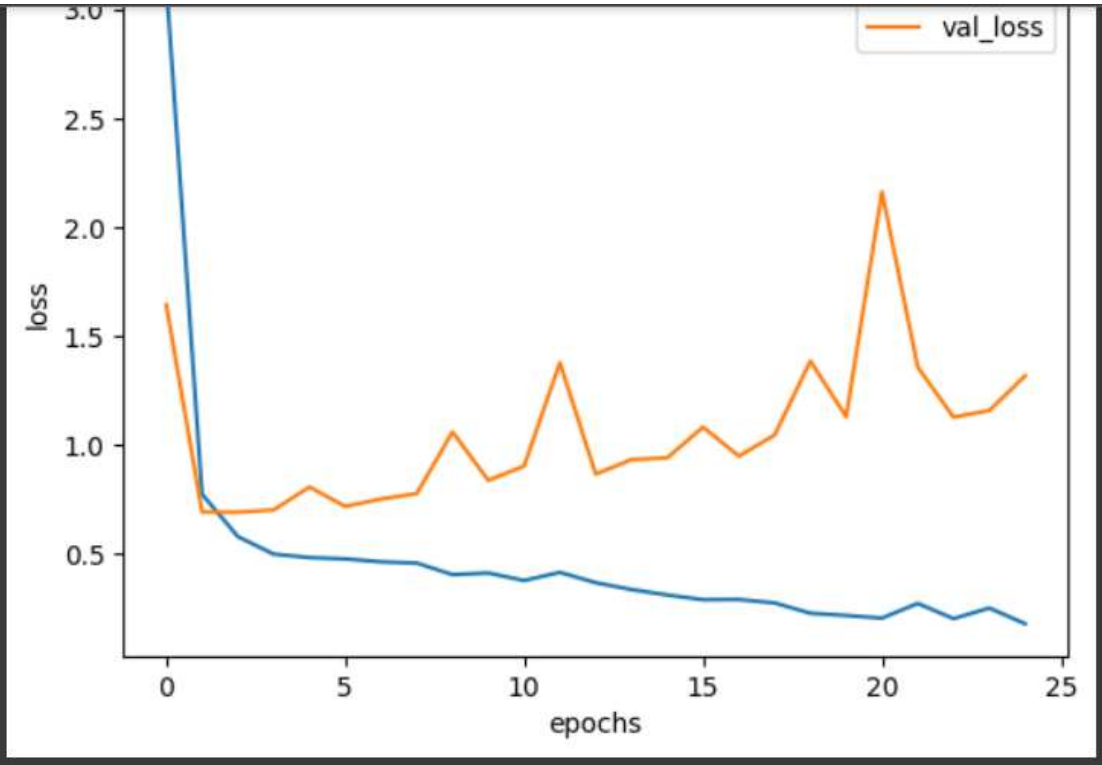
پس از استخراج ویژگی‌ها، از یک مدل ماشین بردار پشتیبان (SVM) برای دسته‌بندی نواحی استفاده شده است.این مرحله برای بهبود دقت طبقه‌بندی و کاهش نرخ خطا انجام شده است.

مدل روی تصاویر آزمایشی اجرا شده است.جعبه‌های محدودکننده پیشنهادی توسط مدل بررسی شده و در صورت تأیید، روی تصاویر رسم شده‌اند.

دقت :(Precision) نسبت نواحی مثبت درست شناسایی شده.

فراخوانی :(Recall) نسبت جعبه‌های واقعی که توسط مدل شناسایی شده‌اند

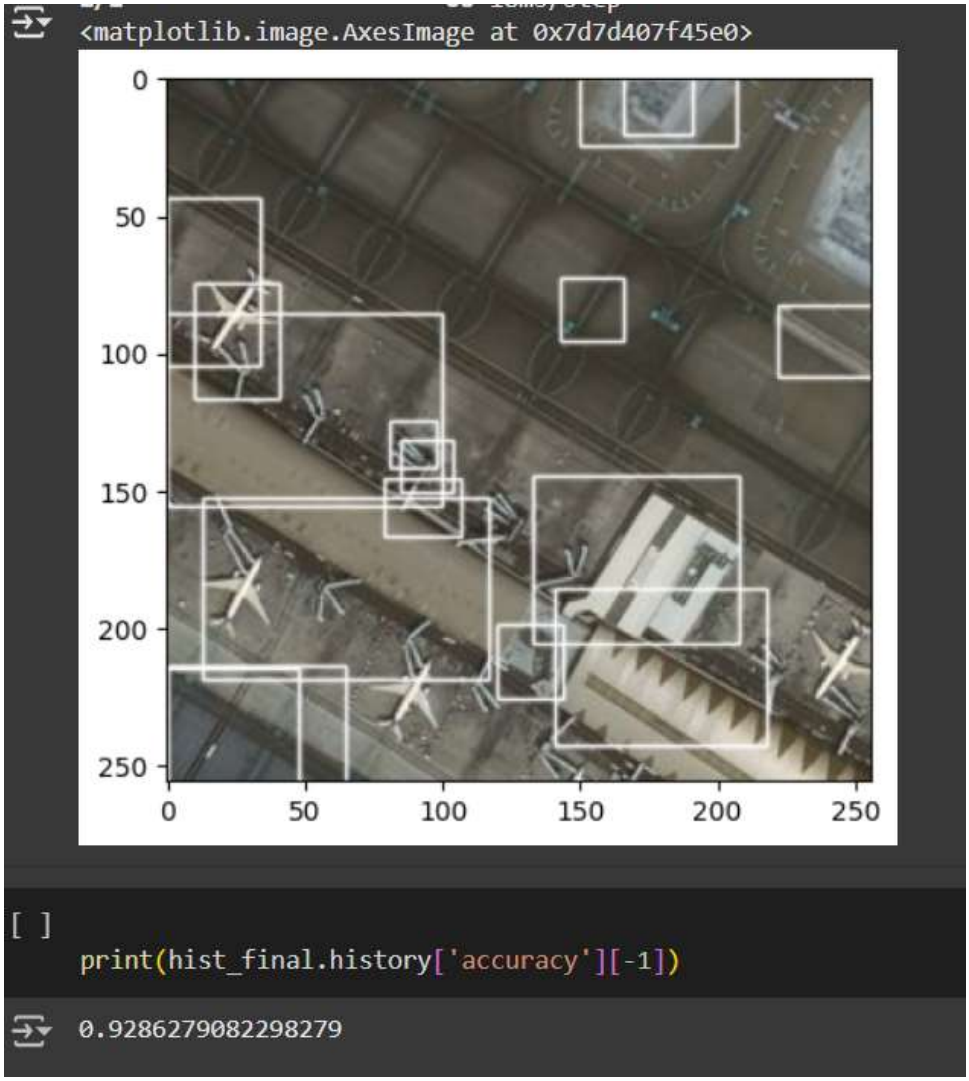
.معیار IoU برای ارزیابی دقیق عملکرد مدل استفاده شده است.



sting the model

```
img_test = cv2.imread(os.path.join(img_path,'42845.jpg'))
sel_search_seg.setBaseImage(img_test)
sel_search_seg.switchToSelectiveSearchFast()
sel_search_seg_res = sel_search_seg.process()

img_out = img_test.copy()
boxes = []
```

```
for e,res in enumerate(sel_search_seg_res):
    if e < 50:
        x,y,w,h = res
        temp_img = img_out[x:x+w,y:y+h]
        resize = cv2.resize(temp_img,(224,224),interpolation= cv2.INTER_AREA)
        resize = np.expand_dims(resize,axis=0)
        out = final_model.predict(resize)
        if (out[0][0]<out[0][1]):
            boxes.append([x,y,w,h])
            count = count+1
for box in boxes:
    x,y,w,h = box
    cv2.rectangle(img_out,(x,y),(x+w,y+h),(255,255,255),1,cv2.LINE_AA)

plt.imshow(img_out)
```



تصاویر با جعبه‌های محدودکننده‌ای که موقعیت هواپیماها را نشان می‌دهند، به‌عنوان خروجی تولید شده‌اند.مدل با دقت بالا توانسته است نواحی مثبت را شناسایی کند و نرخ خطا را به حداقل برساند.

استفاده از (RPN)

یک مدل یادگیری عمیق از پیش‌آموزش‌دیده) مانند VGG16 یا (ResNet به عنوان **شبکه اصلی** (Backbone)استفاده می‌شود.این مدل ویژگی‌های تصویر ورودی را استخراج می‌کند و یک **نقشه ویژگی (Feature Map)** تولید می‌کند.

این نقشه ویژگی ورودی RPN خواهد بود.روی نقشه ویژگی، یک لایه کانولوشن ۳x۳ اعمال می‌شود که خروجی آن ویژگی‌های مکانی برای هر نقطه از نقشه است.این لایه به منظور کاهش پیچیدگی محاسبات و تمرکز روی اطلاعات مکانی طراحی شده است.

برای هر نقطه از نقشه ویژگی، RPNمجموعه‌ای از **جعبه‌های اولیه (Anchor Boxes)** با اندازه‌ها و نسبت ابعاد مختلف تولید می‌کند معمولاً ۹ نوع جعبه اولیه برای هر موقعیت (۳ اندازه و ۳ نسبت ابعاد مختلف) تعریف می‌شود.هر Anchor Box نمایانگر یک ناحیه احتمالی از تصویر است که ممکن است شامل شیء باشد.

برای هر Anchor Box مشخص می‌کند که آیا شامل یک شیء است یا نه Foreground) یا.(Background خروجی این بخش احتمال وجود شیء در Anchor Box است.

مختصات Anchor Box را برای تطبیق بهتر با شیء موجود اصلاح می‌کند.خروجی این بخش شامل تغییرات (dx, dy, dw, dh) برای تنظیم مختصات Anchor Box است.

```
for e, name in enumerate(os.listdir(annotations_path)):
    if name.startswith('airplane'):
        file_name = name.split('.')[0] + '.jpg'
        img = cv2.imread(os.path.join(img_path, file_name))
        dataframe = pd.read_csv(os.path.join(annotations_path, name))
        gt_values = []

        for r in dataframe.iterrows():
            x1, y1, x2, y2 = map(int, r[1][0].split(' '))
            gt_values.append({'x1': x1, 'x2': x2, 'y1': y1, 'y2': y2})

        # Region Proposal using RPN (instead of Selective Search)
        for gval in gt_values:
            temp_img = img[gval['y1']:gval['y2'], gval['x1']:gval['x2']]
            resized = cv2.resize(temp_img, (224, 224), interpolation=cv2.INTER_AREA)
            X_train.append(resized)
            y_train.append(1)

            false_x1 = gval['x1'] - 10 if gval['x1'] > 10 else gval['x1']
            false_y1 = gval['y1'] - 10 if gval['y1'] > 10 else gval['y1']
            false_x2 = gval['x2'] + 10 if gval['x2'] + 10 < img.shape[1] else gval['x2']
            false_y2 = gval['y2'] + 10 if gval['y2'] + 10 < img.shape[0] else gval['y2']

            temp_img = img[false_y1:false_y2, false_x1:false_x2]
            resized = cv2.resize(temp_img, (224, 224), interpolation=cv2.INTER_AREA)
            X_train.append(resized)
            y_train.append(0)
```

این شبکه‌ها که در مدل‌های Faster R-CNN استفاده می‌شوند، جایگزین بهتری برای الگوریتم Selective Search هستند و سرعت و دقت بالاتری دارند.تصاویر متنوع‌تر برای بهبود عملکرد مدل در شرایط مختلف جمع آوری میکنند.

Anchor Boxesبه دو دسته تقسیم می‌شوند؛**مثبت**Anchor **(Positive):** هایی که IoU آن‌ها با جعبه‌های واقعی (Ground Truth)بیشتر از ۰٫۷ باشد.**منفی**Anchor **(Negative):** هایی که IoU آن‌ها با جعبه‌های واقعی کمتر از ۰٫۳ باشد.

تابع هزینه RPN شامل دو بخش است:

هزینه طبقه‌بندی:(Classification Loss) خطای پیش‌بینی مثبت یا منفی بودن.Anchor Box معمولاً از **Cross-Entropy Loss**برای این بخش استفاده می‌شود.

هزینه رگرسیون:(Regression Loss) خطای پیش‌بینی مختصات جعبه توسط شبکه.معمولاً از **Smooth L1 Loss**برای این بخش استفاده می‌شود.

برای جلوگیری از نامتعادلی داده که معمولاًAnchor های منفی بسیار بیشتر ازAnchor های مثبت هستند نمونه‌گیری متعادل انجام می‌شود؛ تعدادAnchor های مثبت و منفی در هر دسته به نسبت ۱:۱ انتخاب می‌شوند (مثلاً ۱۲۸ نمونه مثبت و ۱۲۸ نمونه منفی).

```
def test_model(image_path, model):
    img = cv2.imread(image_path)
    proposals = []

    # Generate Proposals Using RPN
    for _ in range(100):
        x1 = np.random.randint(0, img.shape[1] - 50)
        y1 = np.random.randint(0, img.shape[0] - 50)
        x2 = np.random.randint(x1 + 50, img.shape[1])
        y2 = np.random.randint(y1 + 50, img.shape[0])
        proposals.append((x1, y1, x2, y2))

    img_out = img.copy()

    for x1, y1, x2, y2 in proposals:
        temp_img = img[y1:y2, x1:x2]
        resized = cv2.resize(temp_img, (224, 224), interpolation=cv2.INTER_AREA)
        resized = np.expand_dims(resized, axis=0)
        prediction = model.predict(resized)

        if prediction > 0.5:
            cv2.rectangle(img_out, (x1, y1), (x2, y2), (0, 255, 0), 2)

    plt.imshow(cv2.cvtColor(img_out, cv2.COLOR_BGR2RGB))
    plt.show()
```

```
test_model('/content/gdrive/My Drive/airplanes/Images/Images/Planes8.jpg', model)
```

```python
from keras.layers import Dense, Flatten
from keras.applications.vgg16 import VGG16

vgg = VGG16(include_top=False, weights='imagenet', input_shape=(224, 224, 3))
x = Flatten()(vgg.output)
x = Dense(128, activation='relu')(x)
x = Dense(1, activation='sigmoid')(x)
model = Model(vgg.input, x)

for layer in vgg.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```python
from keras.optimizers import Adam

model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])

# Training the Model
model.fit(X_train, y_train, batch_size=32, epochs=10, validation_split=0.1, shuffle=True)
```

چرخش، تغییر مقیاس و دیگر تغییرات روی تصاویر برای افزایش تنوع داده‌ها اعمال میشود.تنظیم دقیق مدل برای کاهش نواحی‌ای که به اشتباه به‌عنوان مثبت شناسایی شده‌اند .

```
true_labels = []
predicted_labels = []

for e, res in enumerate(sel_search_seg_res):
    if e < 50:
        x, y, w, h = res
        temp_img = img_out[y:y + h, x:x + w]
        resize = cv2.resize(temp_img, (224, 224), interpolation=cv2.INTER_AREA)
        resize = np.expand_dims(resize, axis=0)

        out = final_model.predict(resize)
        predicted_class = np.argmax(out[0])
        predicted_labels.append(predicted_class)

        true_labels.append(1 if IOU({'x1': x, 'x2': x + w, 'y1': y, 'y2': y + h

accuracy = accuracy_score(true_labels, predicted_labels)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
Accuracy: 58.00%
```

استفاده از شبکه‌های YOLO یا SSDبرای شناسایی بلادرنگ.