

به نام خدا



دانشگاه تهران



دانشکده مهندسی برق و کامپیوتر

درس هوش مصنوعی قابل اعتماد

تمرین دوم

محمد رضا سلیمی

نام و نام خانوادگی

810102178

شماره دانشجویی

فهرست

1.....	پاسخ ۱. تفسیر پذیری داده های جدولی
1.....	بخش اول: بار گذاری داده و آموزش مدل
1.....	بار گذاری داده
7.....	آموزش و ارزیابی مدل
9.....	بخش دوم: تفسیر مدل
9.....	روش LIME
13.....	روش SHAP
23.....	بخش سوم : مدل NAM
33.....	پاسخ ۲ - تفسیرپذیری در حوزه تصویر
33.....	GRAD-CAM-1
33.....	1 – 1
34.....	2 – 1
37.....	GUIDED GRAD-CAM
37.....	1 -2
38.....	2 – 2
42.....	3 – 2
43.....	4 – 2
47.....	SMOOTHGRAD
47.....	1 – 3
48.....	2-3
53.....	3 -3
56.....	ADVERSARIAL PETREBUTION AND PIXEL ATTRIBUTION
56.....	1-4
58.....	FEATURE VISUALIZATION
58.....	1
59.....	2

60.....	3
62.....	4

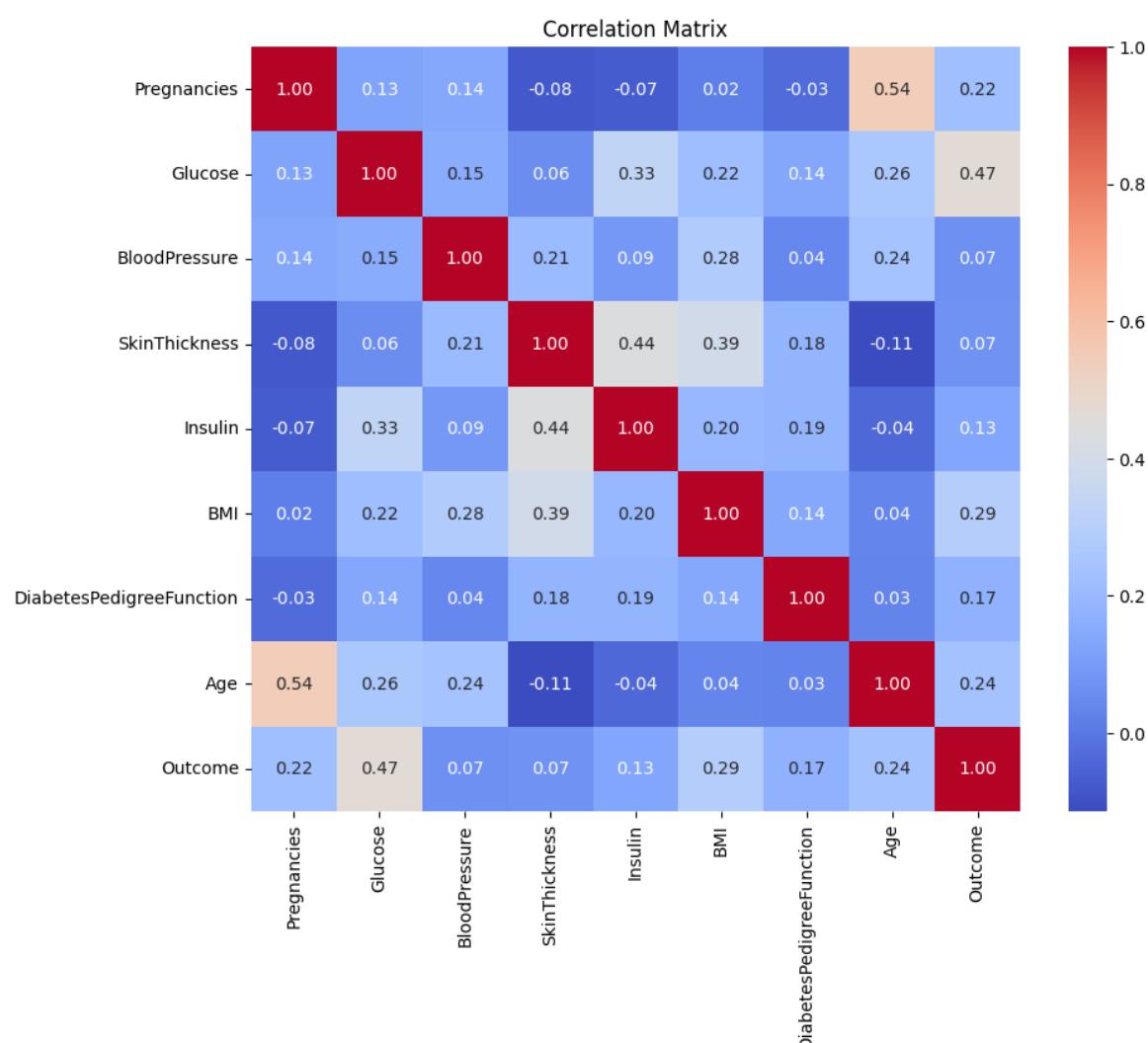
پاسخ 1. تفسیر پذیری داده های جدولی

بخش اول: بار گذاری داده و آموزش مدل

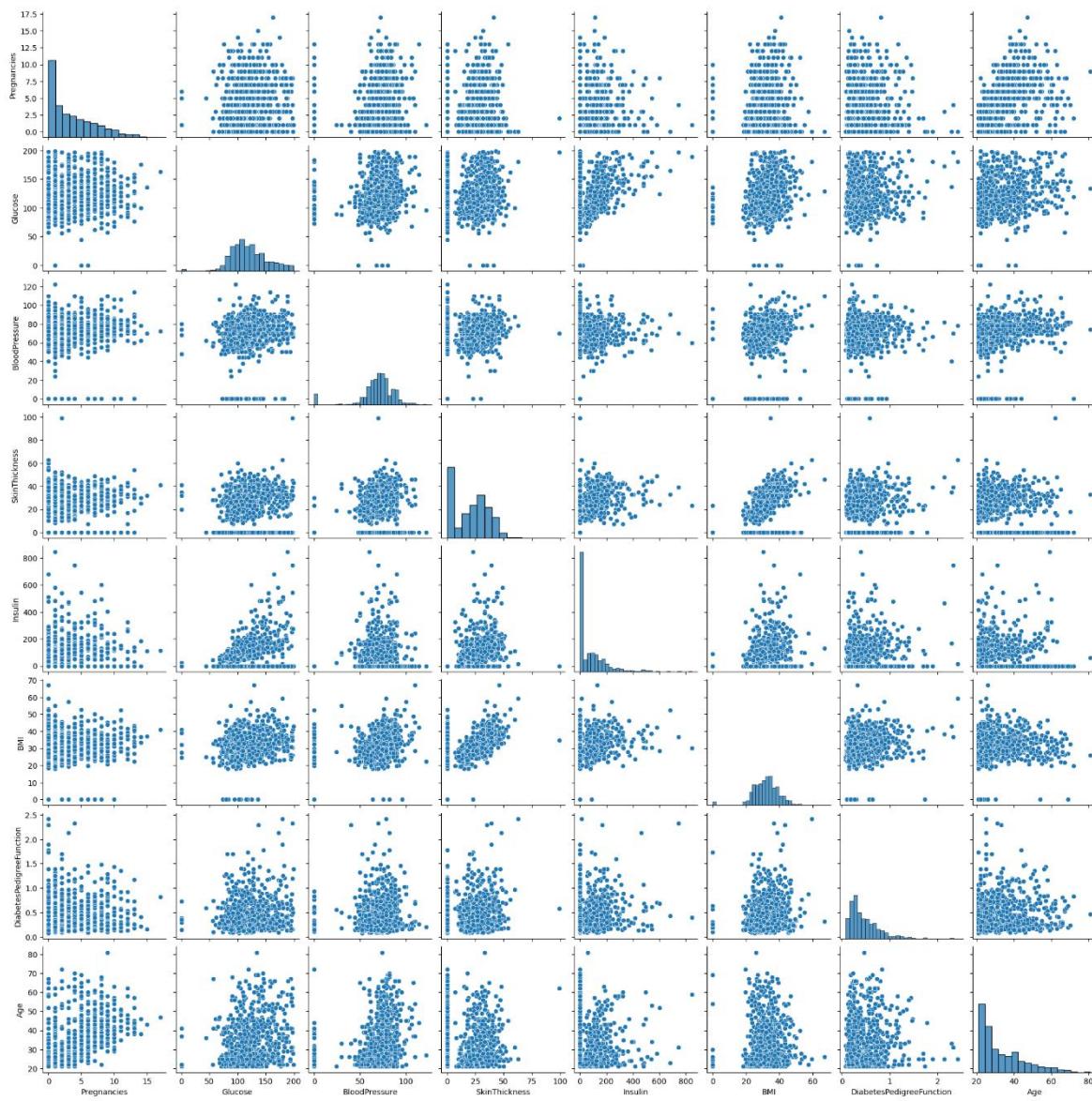
بار گذاری داده

(1

الف : ابتدا ماتریس وابستگی (Correlation matrix) و نمودار Pair plot داده ها را رسم کردم که به این شکل شد .



ماتریس وابستگی داده ها



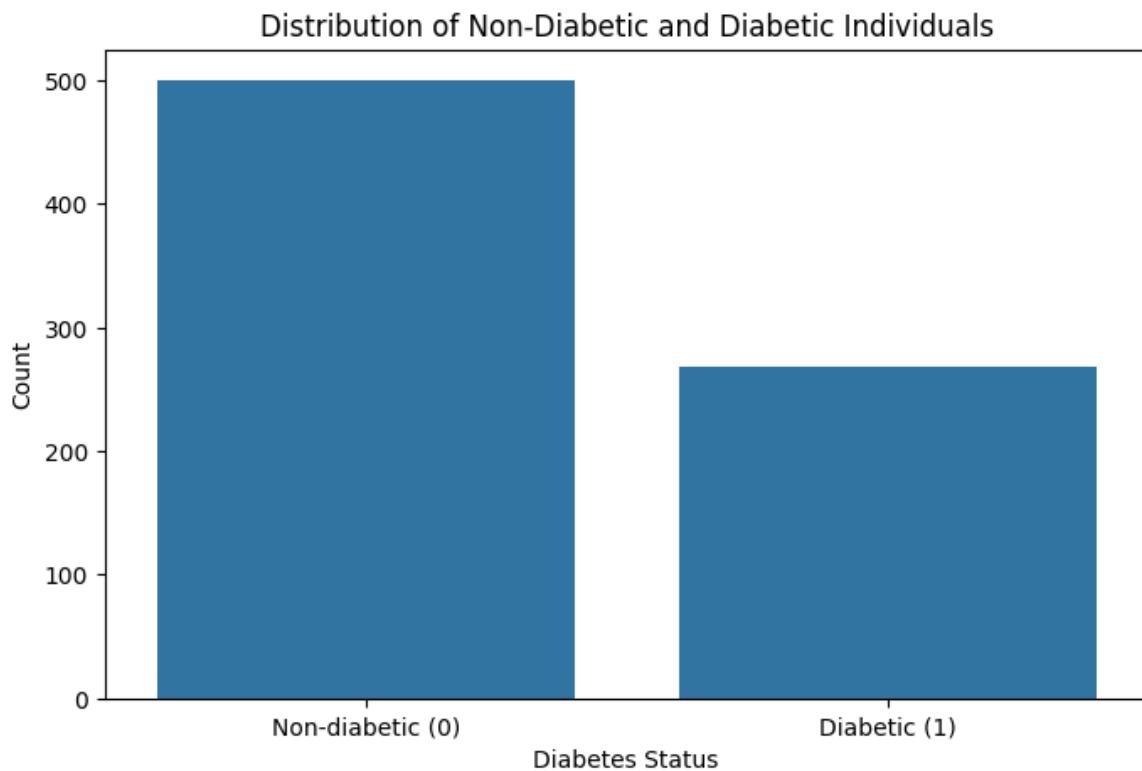
نمودار Pair plot

تحلیل :

طبق نمودار وابستگی ، ضریب وابستگی بین Age و Pregnancies 0.54 است که نشان دهنده یک وابستگی قوی مثبت بین این دو متغیر است که این را از نمودار Pair plot هم میتوان مشاهده کرد .

طبق ماتریس وابستگی ضریب وابستگس بین SkinThickness و BMI برابر 0.39 است که نشان دهنده وابستگی متوسط مثبتی است. از نمودار pairplot مشاهده می‌شود که این دو ویژگی رابطه خطی نسبتاً مستقیمی دارند، به این معنی که با افزایش SkinThickness، BMI نیز معمولاً افزایش می‌یابد

ب



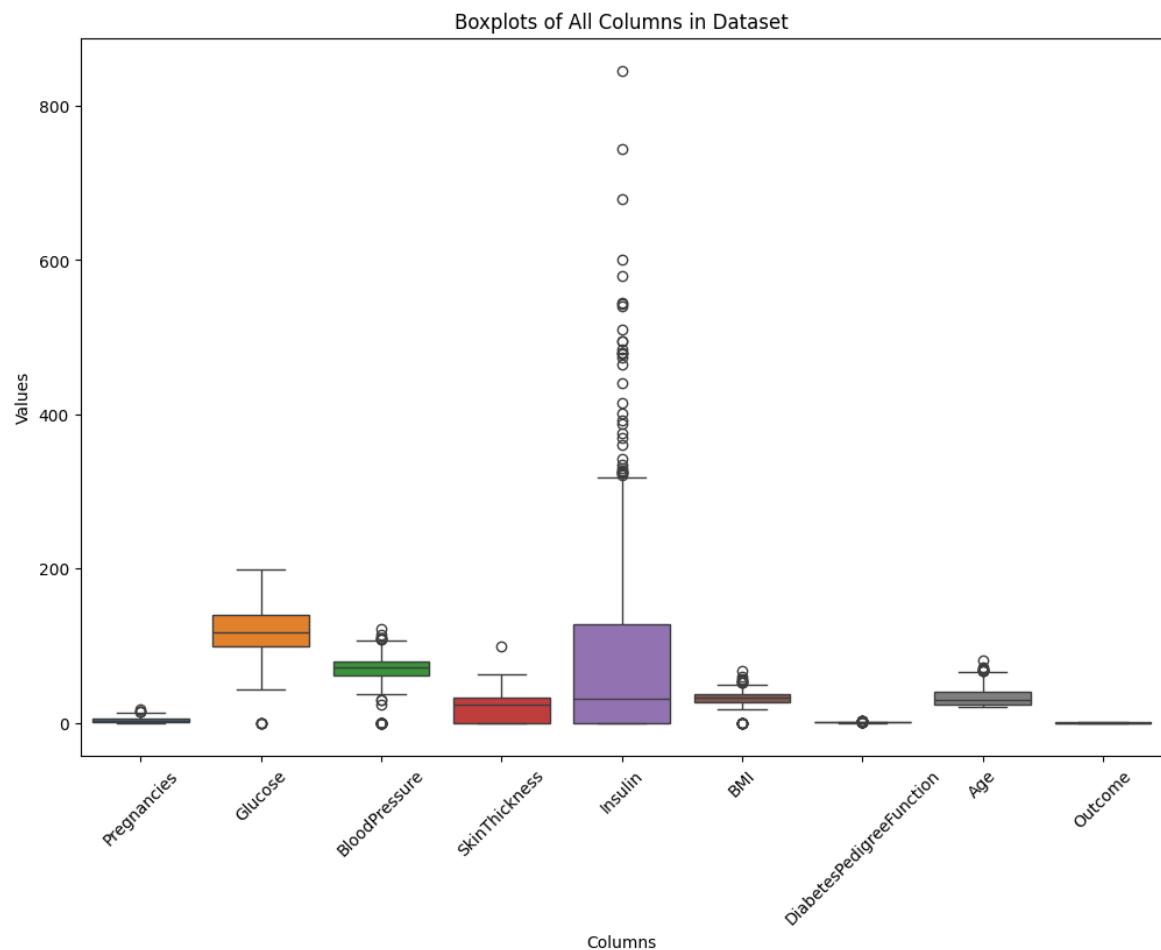
نمودار توزیع افراد سالم و افراد مبتلا به دیابت

تحلیل :

طبق نمودار تعداد افراد غیر دیابتی (Non-diabetic)، نسبت به افراد دیابتی (Diabetic) است این نمودار نشان می‌دهد که تعداد افراد غیر دیابتی تقریباً بیش از دو برابر افراد دیابتی است، که می‌تواند نشان دهنده نامتوازن بودن داده‌ها در زمینه وضعیت دیابت باشد.

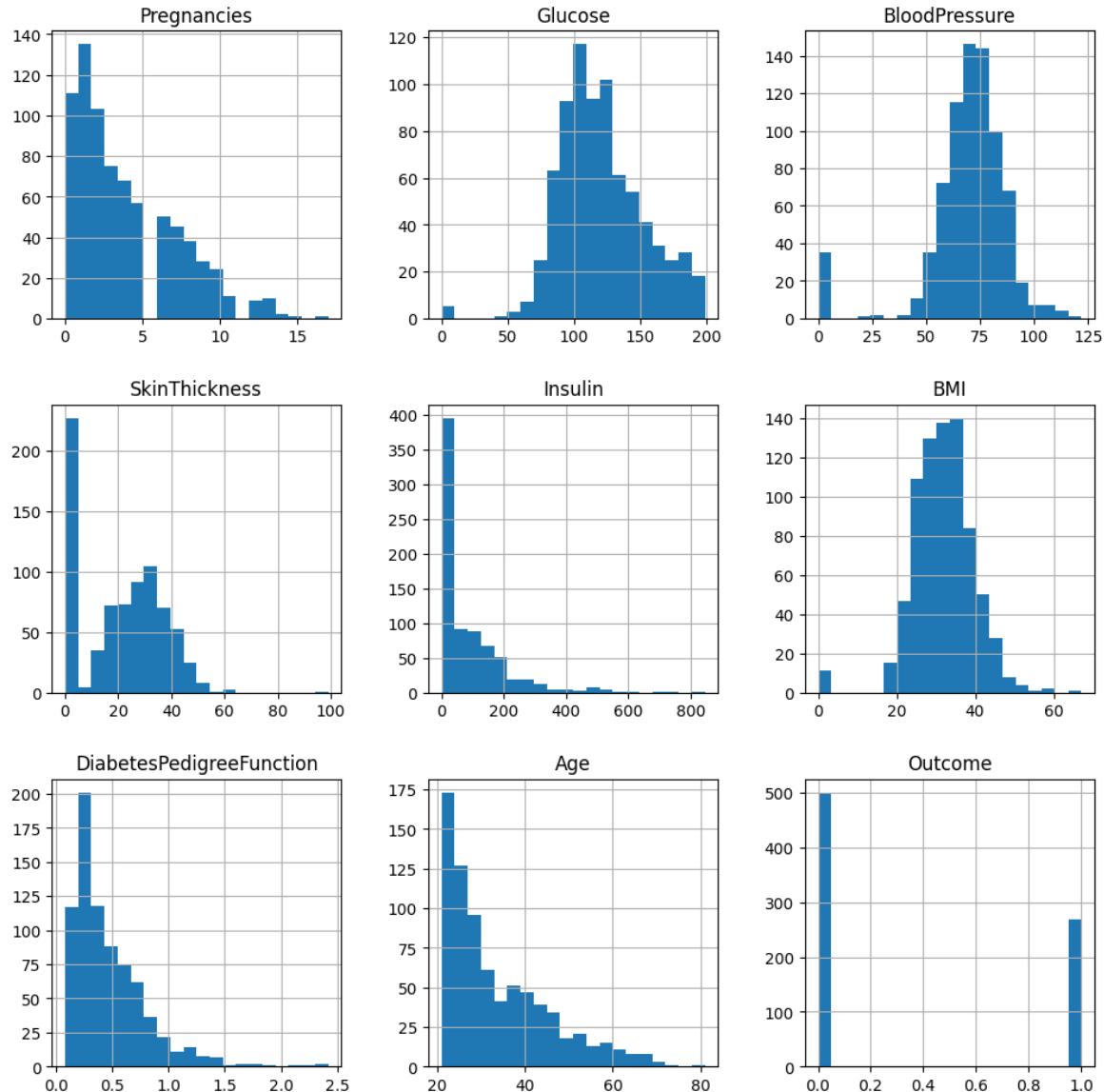
: پ

برای تحلیل داده های پرت از نمودار جعبه ای (Box plot) و نمودار هیستوگرام استفاده کرده ام.



نمودار جعبه ای داده ها

Histograms of All Columns in Dataset



نمودار هیستوگرام داده ها

تحلیل :

طبق نمودار جعبه ای Insulin دارای تعداد زیادی داده پرت است که مقادیر بسیار بالاتری نسبت به میانه دارند. همچنین مقادیر زیاد داده های پرت در ستون SkinThickness نیز مشاهده میشود .

در BMI و BloodPressure نیز چند داده پرت وجود دارد .

نتایج بالا را میتوان در نمودار هیستوگرام نیز مشاهده کرد . طبق هیستوگرام Insulin و SkinThickness دارای توزیعهایی با دنبالههای طولانی به سمت ارزشهای بالاتر هستند، که نشاندهنده وجود دادههای پرت است. همچنین BMI و BloodPressure نیز نشاندهنده پراکندگی متغیر با دادههای پرت در دو انتهای توزیع هستند.

اگر تعداد داده های پرت زیاد باشد باعث تاثیر منفی بر روی دقت و کارایی مدل میشود زیرا دادههای پرت ممکن است سبب شوند که مدل نتواند الگوهای دقیقی را برای دادههای عمومی تر یاد بگیرد.

همچنین در تحلیلهای طبقهبندی مانند تحلیل ما، وجود دادههای پرت میتواند باعث شود که مدل در پیش‌بینی دادههای جدید دچار خطا شود.

2

در کد برای اینکه دقت بالاتر رود ابتدا پیش پردازش های زیر را انجام داده ام در ابتدا مقادیر صفر BMI و Glucose را حذف کرده ام زیرا مقادیر صفر این دو داده اندک هست و در واقعیت این دو معیار نمیتوانند صفر باشند همچنین طبق box plot مقادیر صفر این دو معیار جز داده های پرت محسوب میشود

در ادامه داده ها را با StandardScaler نرمال کرده ام همچنین چون توزیع افراد دیابتی و غیر دیابتی خیلی با هم متفاوت است و افراد غیر دیابتی تقریباً دو برابر افراد دیابتی هستند دو کلاس را با SMOTE متوازن کرده ام . برای متوازن کردن over sampling استفاده کردم و داده های کلاس کمتر یعنی افراد غیر دیابتی را به صورت مصنوعی افزایش داده ام .

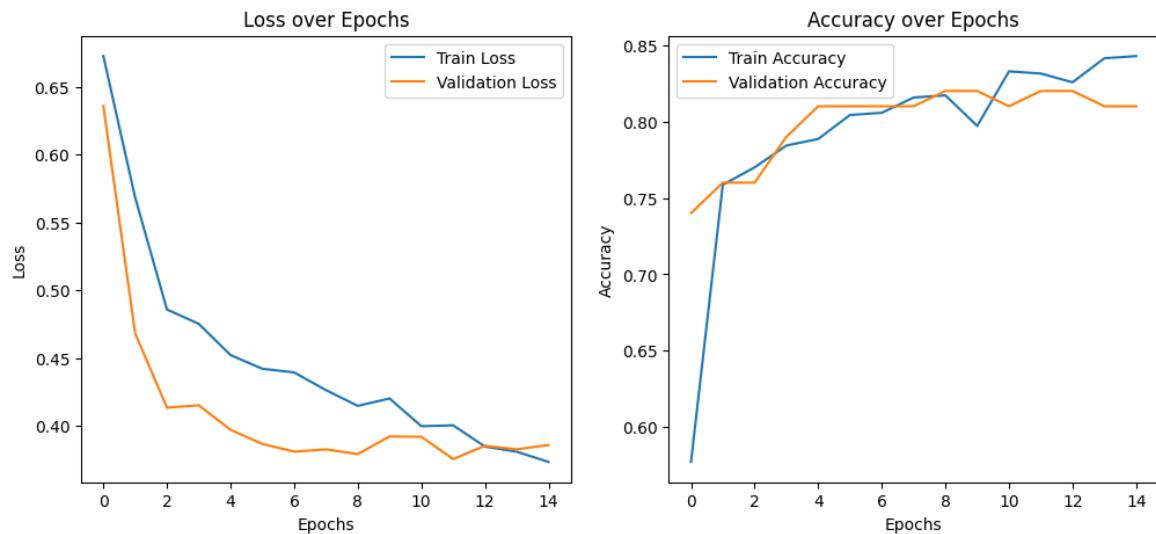
در ادامه دادها را به سه دسته آموزش ارزیابی و تست به نسبت 70 و 10 و 20 تقسیم کردم .

معماری مدل نظر را طبق جدول داده شده پیاده سازی کردم

```
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.layer1 = nn.Linear(8, 100)
        self.bn1 = nn.BatchNorm1d(100)
        self.layer2 = nn.Linear(100, 50)
        self.dropout = nn.Dropout(0.2)
        self.layer3 = nn.Linear(50, 50)
        self.layer4 = nn.Linear(50, 20)
        self.output_layer = nn.Linear(20, 1)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.relu(self.layer1(x))
        x = self.bn1(x)
        x = self.relu(self.layer2(x))
        x = self.dropout(x)
        x = self.relu(self.layer3(x))
        x = self.relu(self.layer4(x))
        x = torch.sigmoid(self.output_layer(x))
        return x
```

همچنین داده ها در 15 ایپاک با بهینه ساز آدام آموزش داده ام که نمودار تغییرات خطای ودقت برای داده های آموزشی و ارزیابی به این شکل شد .



نمودار تغییرات خطای داده های آموزش و ارزیابی

همچنین مدل را روی داده های تست ارزیابی کرده که نتیجه به این شکل شد.

معیار	مقدار
(Accuracy) دقت	0.8
Precision	0.7248
Recall	0.8876
F1 (F1 Score) امتیاز	0.798

همچنین Confusion matrix اینگونه شد

Confusion Matrix:

$\begin{bmatrix} 81 & 30 \end{bmatrix}$

$\begin{bmatrix} 10 & 79 \end{bmatrix}$

بخش دوم: تفسیر مدل

LIME روش

در این قسمت برای سه نمونه از داده های تست را به صورت رندوم انتخاب کرده ام و به روش Lime از کتابخانه LimeTabularExplainer تحلیل های مناسب را انجام داده ام .
داده های تست شماره 163 و 28 و 6 انتخاب شده اند که لیل واقعی آنها به ترتیب 1 و 0 و 1 است و مدل هم دقیقا همین لیبل ها را به درستی پیش بینی کرده است .

ابتدا داده تست شماره 163 را بررسی میکنیم (توجه شود مقادیری که در اینجا آمده است مقادیر نرمال شده است)



نمودار Lime داده شماره 163 تست

در اینجا، مدل پیش بینی کرده است که فرد مورد نظر به احتمال 54 درصد مبتلا به دیابت است و لیبل 1 را به فرد داده است که درست هم پیش بینی کرده است .

تحلیل ویژگی ها و تاثیر آنها بر پیش بینی مدل

BMI > 0.58 (0.219)

تأثیر مثبت: این ویژگی بیشترین تأثیر مثبت را بر پیش بینی دیابت دارد. با توجه به نمودار، BMI بیشتر از 0.58 به میزان 0.219 واحد به احتمال ابتلا به دیابت اضافه کرده است.

DiabetesPedigreeFunction > 0.47 (0.182)

تأثیر مثبت: این ویژگی بیشترین تأثیر مثبت را بر پیش‌بینی دیابت دارد. با توجه به نمودار، BMI بیشتر از 0.58 به میزان 0.219 واحد به احتمال ابتلا به دیابت اضافه کرده است.

-0.12 < Glucose < 0.61 (0.089)

تأثیر مثبت: مقدار Glucose در بازه 0.12- 0.61 نیز به میزان 0.089 واحد به احتمال دیابت اضافه کرده است.

Age < -0.79 (-0.075):

تأثیر منفی: سن کمتر یا مساوی -0.79 تأثیر منفی بر پیش‌بینی دیابت داشته و 0.075 واحد از احتمال دیابت کاسته است.

-0.84 < Pregnancies < -0.25 (-0.065):

تأثیر منفی: تعداد بارداری‌ها در بازه -0.25- 0.84 نیز به میزان 0.065 واحد از احتمال دیابت کاسته است.

BloodPressure > 0.56 (0.037):

تأثیر مثبت: فشار خون بیشتر از 0.56 تأثیر مثبتی داشته و 0.037 واحد به احتمال دیابت اضافه کرده است.

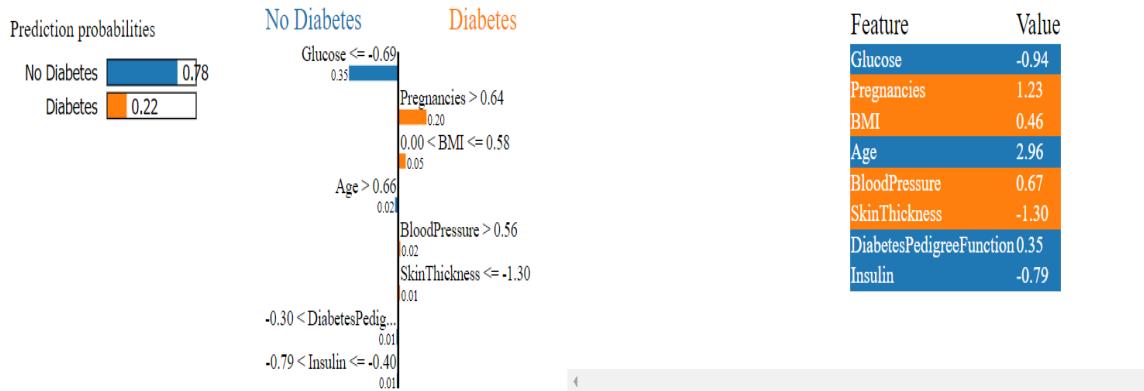
-0.79 < Insulin < -0.40 (0.024):

تأثیر مثبت: مقدار انسولین در بازه -0.79- 0.40 تأثیر مثبتی داشته و 0.024 واحد به احتمال دیابت اضافه کرده است.

SkinThickness <= -1.30 (-0.002):

تأثیر منفی: ضخامت پوست کمتر یا مساوی -1.30 تأثیر منفی جزئی داشته و 0.002 واحد از احتمال دیابت کاسته است.

داده تست شماره 28 :

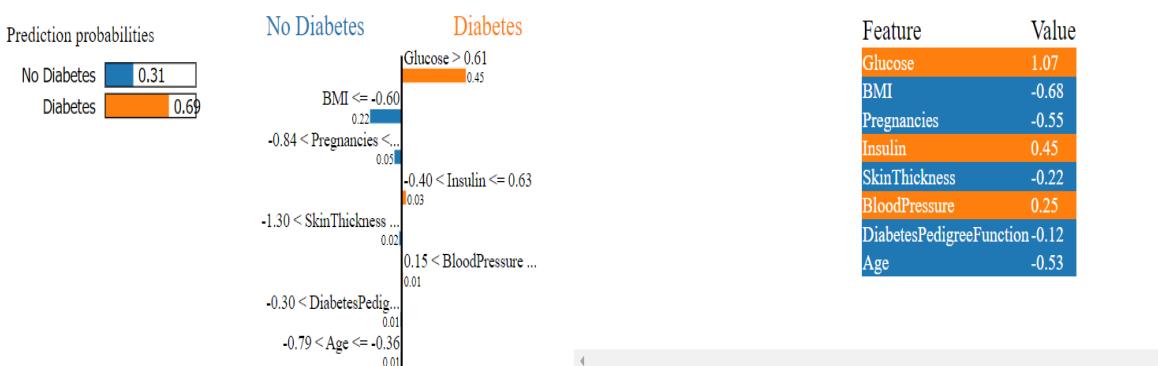


نمودار Lime داده شماره 28 تست

طبق تصویر مدل به احتمال 78 درصد تشخیص داده است که فرد مورد نظر دیابتی نیست که با توجه به لیبل های واقعی درست هم تشخیص داده است.

طبق تصویر مقادیر پایین گلوکز و سن بالا تأثیر بسیار مثبتی در کاهش احتمال دیابت دارند. ویژگی های دیگری مانند DiabetesPedigreeFunction و انسولین نیز تأثیر مثبتی بر عدم ابتلا به دیابت دارند، اما تأثیر آنها کمتر است. از سوی دیگر، تعداد بارداری ها و مقادیر BMI، فشار خون و ضخامت پوست تأثیر منفی داشته و احتمال دیابت را افزایش می دهند، اگرچه تأثیر آنها ضعیفتر است.

داده تست شماره 6 :



نمودار Lime داده شماره 6 تست

تحلیل :

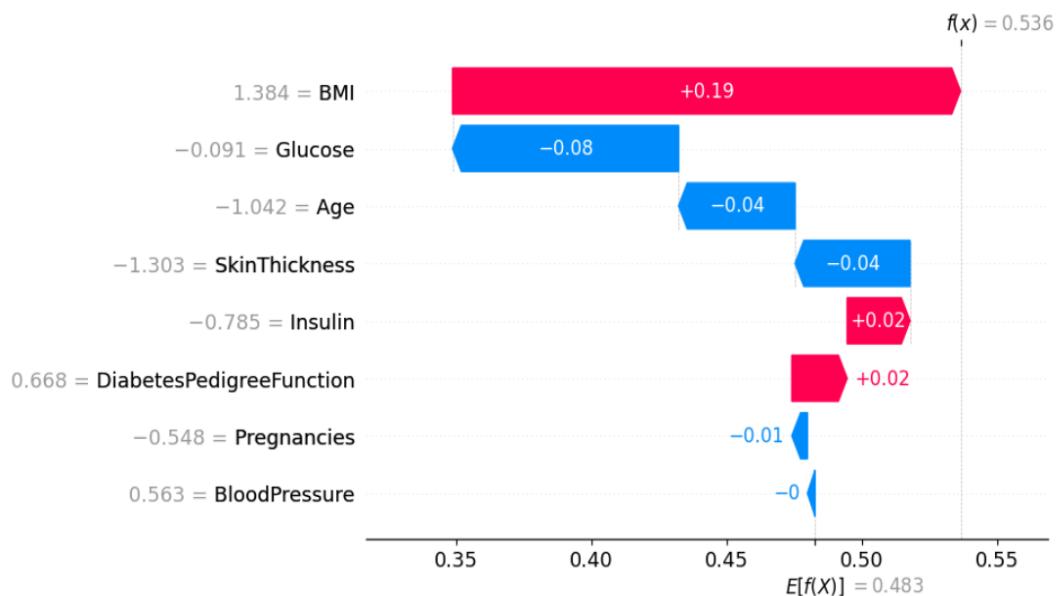
نمونه به احتمال 69 درصد پیش بینی کرده است که نمونه مورد نظر دیابتی است که با توجه به لیل های واقعی درست پیش بینی کرده است.

همچنین طبق نمودار گلوکز بالا تأثیر بسیار زیادی بر افزایش احتمال دیابت دارد BMI . پایین نیز تأثیر بسیار زیادی بر کاهش احتمال دیابت دارد. ویژگی‌های دیگری مانند تعداد بارداری‌ها و انسولین نیز تأثیر مثبتی بر پیش‌بینی دیابت دارند، اما تأثیر آنها کمتر است.

روش SHAP

در این قسمت میخواهیم نمودار Force Plot که از کتابخانه SHAP تولید شده است را برای همان سه نمونه قبلی تحلیل کنیم

نمودار Force plot مربوط به نمونه تست شماره 163 :



نمودار Force plot مربوط به نمونه تست شماره 163 :

تحلیل :

BMI (1.384) [+0.19]:

تأثیر مثبت: این ویژگی بیشترین تأثیر مثبت را بر پیش‌بینی دیابت داشته است. مقدار 1.384 برای BMI باعث افزایش 0.19 در پیش‌بینی نهایی شده است.

DiabetesPedigreeFunction (0.668) [+0.02]:

تأثیر مثبت: Diabetes Pedigree Function با مقدار 0.668 به میزان 0.02 به پیش‌بینی دیابت اضافه کرده است.

BloodPressure (0.563) [+0.02]:

تأثیر مثبت: مقدار فشار خون برابر 0.563 نیز تأثیر مثبتی داشته و 0.02 واحد به احتمال دیابت اضافه کرده است.

Insulin (-0.785) [+0.02]:

تأثیر مثبت: مقدار انسولین کمتر از متوسط (-0.785) به میزان 0.02 واحد به احتمال دیابت اضافه کرده است.

Glucose (-0.091) [-0.08]:

تأثیر منفی: مقدار گلوکز منفی (-0.091) باعث کاهش 0.08 در پیش‌بینی دیابت شده است.

Age (-1.042) [-0.04]:

تأثیر منفی: سن پایین‌تر از میانگین (-1.042) باعث کاهش 0.04 در پیش‌بینی دیابت شده است.

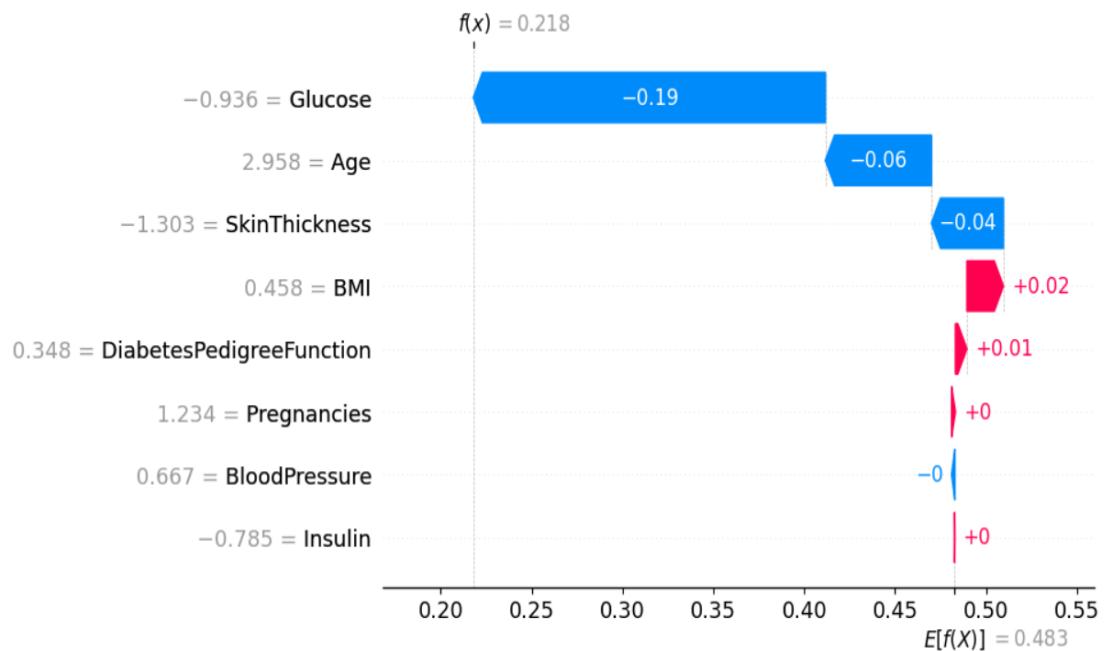
SkinThickness (-1.303) [-0.04]:

تأثیر منفی: ضخامت پوست کمتر از متوسط (-1.303) به میزان 0.04 واحد از احتمال دیابت کاسته است.

Pregnancies (-0.548) [-0.01]:

تأثیر منفی: تعداد بارداری‌ها کمتر از میانگین (-0.548) تأثیر منفی کوچکی داشته و 0.01 واحد از احتمال دیابت کاسته است.

داده تست شماره 28 :

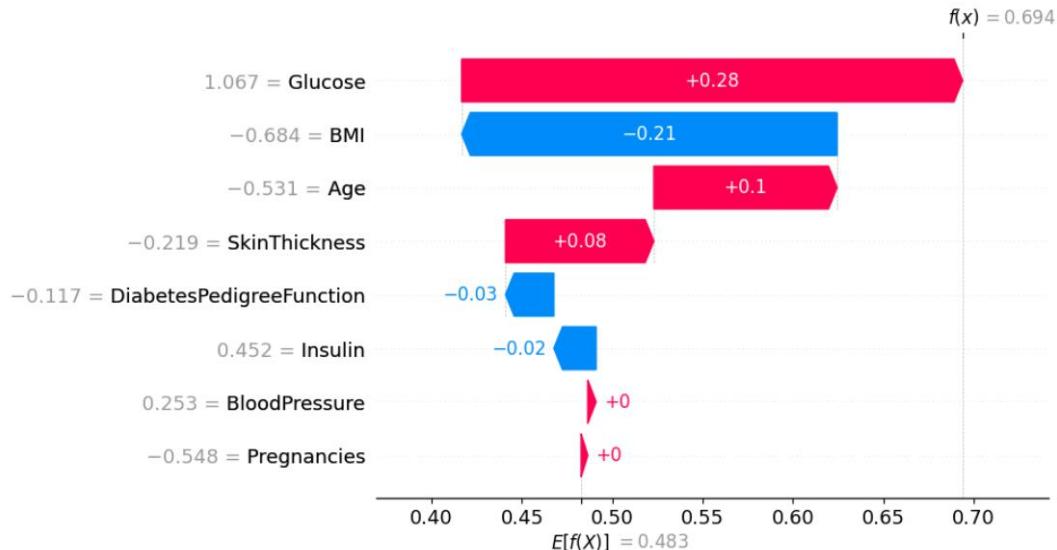


نمودار Force plot مربوط به نمونه تست شماره 28

تحلیل :

طبق نمودار احتمال ابتلا به دیابت برای این نمونه 21.8 درصد است (طبق $f(x)$) همچنین نمودار نشان میدهد که مقدار گلوکز پایین (-0.936) بیشترین تأثیر منفی را بر احتمال ابتلا به دیابت دارد. سن بالا (2.958) و ضخامت پوست پایین (-1.303) نیز تأثیر منفی قابل توجهی بر کاهش احتمال دیابت دارند. BMI مقدار کمی مثبت تأثیر دارد و دیگر ویژگی‌ها مانند Diabetes Pedigree Function، تعداد بارداری‌ها، فشار خون و انسولین تأثیر بسیار کمی دارند.

نمونه تست شماره 6 :



نمودار Force plot مربوط به نمونه تست شماره 6

طبق تصویر مدل به احتمال 69.4 درصد پیش بینی کرده است که فرد مورد نظر دیابت دارد.

همچنین نشان میدهد که مقدار گلوکز بالا (1.067) تأثیر بسیار زیادی بر افزایش احتمال دیابت دارد و مهم‌ترین ویژگی است که بر پیش‌بینی دیابت تأثیر می‌گذارد. BMI پایین (کمتر از -0.684) تأثیر زیادی بر کاهش احتمال دیابت دارد. سن و ضخامت پوست نیز تأثیر مثبتی بر افزایش احتمال دیابت دارند، اگرچه تأثیر آنها کمتر است. ویژگی‌های DiabetesPedigreeFunction و انسولین تأثیر منفی ملایمی دارند و Pregnancies و BloodPressure تأثیر بسیار ضعیفی بر پیش‌بینی دیابت دارند.

الف :

با توجه به نمونه‌ها و نمودارها ، می‌توان برخی از شباهت‌ها و تفاوت‌های بین روش‌های LIME و SHAP را به طور کلی مورد بررسی قرار داد.

شباهت‌ها:

تأثیر مثبت و منفی ویژگی‌ها:

هر دو روش LIME و SHAP نشان می‌دهند که چگونه ویژگی‌های مختلف تأثیر مثبت یا منفی بر پیش‌بینی مدل دارند.

در هر دو روش، ویژگی‌هایی مانند Glucose و BMI به طور معمول به عنوان مهم‌ترین ویژگی‌ها شناسایی می‌شوند.

شناسایی ویژگی‌های مهم:

هر دو روش معمولاً ویژگی‌هایی که بیشترین تأثیر را بر پیش‌بینی دارند، شناسایی می‌کنند. برای مثال، Glucose در هر دو روش تأثیر قابل توجهی دارد.

قابلیت تفسیر محلی:

هر دو روش قابلیت تفسیر محلی را دارند و می‌توانند تأثیر هر ویژگی بر پیش‌بینی یک نمونه خاص را نشان دهند.

تفاوت‌ها :

پیچیدگی محاسباتی:

SHAP: محاسبات پیچیده‌تری دارد و ممکن است زمان بیشتری برای محاسبه نیاز داشته باشد، به ویژه برای مدل‌های بزرگ و پیچیده.

LIME: محاسبات ساده‌تری دارد و برای مدل‌های بزرگ و پیچیده سریع‌تر است.

نحوه ارائه تأثیرات:

LIME: تأثیر ویژگی‌ها را به صورت ساده و قابل فهم ارائه می‌دهد. مدل‌های محلی ساده‌ای را می‌سازد که به راحتی قابل درک هستند.

: تأثیر ویژگی‌ها را به صورت دقیق و عددی ارائه می‌دهد. نمودارهای SHAP می‌توانند تأثیر هر ویژگی را به صورت دقیق و قابل تفسیر نشان دهند.

: ب

برای بررسی دقت تفسیر پذیری دو مدل میتوانیم این دو نمونه را در نظر بگیریم

: نمونه 1

: LIME

$\text{Glucose} > 0.61$: +0.45

$\text{BMI} \leq -0.60$: -0.22

تأثیرات دیگر ویژگی‌ها با دقت کمتری مشخص شده‌اند.

: SHAP

Glucose : +0.28

BMI : -0.21

تأثیر دقیق هر ویژگی به صورت عددی مشخص شده است.

: نمونه 2

: LIME

$\text{BMI} > 0.58$: +0.22

$\text{DiabetesPedigreeFunction} > 0.47$: +0.18

$\text{Glucose} \leq -0.69$: -0.35

: SHAP

Glucose : -0.19

BMI : +0.02

Age : -0.06

نتیجه‌گیری:

SHAP به دلیل دقیق‌تر در محاسبه تأثیرات ویژگی‌ها و استفاده از نظریه بازی‌ها، معمولاً تفسیر دقیق‌تری ارائه می‌دهد.

LIME سریع‌تر و ساده‌تر است و برای تفسیرهای سریع و محلی مناسب است، اما ممکن است دقیق‌تر کمتری داشته باشد.

در نتیجه با توجه به نمونه‌های مشاهده شده، SHAP معمولاً دقیق‌تر در تفسیر اهمیت ویژگی‌ها دارد و می‌تواند به طور دقیق‌تر نشان دهد که کدام ویژگی‌ها بیشترین تأثیر را دارند. از سوی دیگر، می‌تواند برای تفسیر سریع و محلی مناسب باشد و دید کلی از تأثیرات ویژگی‌ها ارائه دهد.

: ج

بررسی ارتباط میان **SHAP** ، **LIME** و ماتریس همبستگی

:Glucose

:LIME

Glucose > 0.61: +0.45

:SHAP

Glucose: +0.28

ماتریس همبستگی: ضریب همبستگی Glucose با Outcome برابر 0.49 است که نشان می‌دهد Glucose رابطه مثبت و قوی‌ای با Outcome دارد. هر دو روش SHAP و LIME تأثیر مثبت زیادی برای Glucose در پیش‌بینی دیابت نشان می‌دهند که با ماتریس همبستگی همخوانی دارد.

:BMI

:LIME

BMI <= -0.60: -0.22

:SHAP

BMI: -0.21

ماتریس همبستگی: ضریب همبستگی Outcome BMI با نشان‌دهنده رابطه مثبت متوسط است. هر دو روش تأثیر منفی برای مقدار پایین BMI در کاهش احتمال دیابت نشان می‌دهند که با همبستگی مثبت میان Outcome و BMI سازگار است.

:Age

:LIME

Age > 0.66: +0.22

:SHAP

Age: +0.10

ماتریس همبستگی: ضریب همبستگی Age با نشان‌دهنده رابطه مثبت ضعیف است. هر دو روش تأثیر مثبت ضعیف تا متوسطی برای Age در افزایش احتمال دیابت نشان می‌دهند که با ماتریس همبستگی سازگار است.

:SkinThickness

:LIME

SkinThickness <= 0.16: -0.02 > 1.30-

:SHAP

SkinThickness: +0.08

ماتریس همبستگی: ضریب همبستگی SkinThickness با نشان‌دهنده رابطه مثبت بسیار ضعیف است. هر دو روش تأثیرات کم اما متناقض برای این ویژگی نشان می‌دهند.

:Insulin

:LIME

Insulin <= 0.63: +0.03 > 0.40-

:SHAP

Insulin: -0.02

ماتریس همبستگی: ضریب همبستگی Insulin با Outcome برابر 0.13 است که نشان‌دهنده رابطه مثبت ضعیف است. تأثیر Insulin در LIME مثبت و در SHAP منفی است، که نشان‌دهنده اختلاف در تفسیر این ویژگی توسط دو روش است.

:DiabetesPedigreeFunction

:LIME

DPF ≤ 0.47 : -0.01 > 0.30 -

:SHAP

DiabetesPedigreeFunction: -0.03

ماتریس همبستگی: ضریب همبستگی DPF با Outcome برابر 0.18 است که نشان‌دهنده رابطه مثبت ضعیف است. هر دو روش تأثیر منفی ضعیفی برای این ویژگی در کاهش احتمال دیابت نشان می‌دهند.

:BloodPressure

:LIME

BloodPressure ≤ 0.56 : +0.01 > 0.15

:SHAP

BloodPressure: 0.00

ماتریس همبستگی: ضریب همبستگی BloodPressure با Outcome برابر 0.05 است که نشان‌دهنده رابطه بسیار ضعیف است. هر دو روش تأثیرات ناچیزی برای این ویژگی در پیش‌بینی دیابت نشان می‌دهند.

:Pregnancies

:LIME

Pregnancies ≤ -0.25 : -0.05 > 0.84 -

:SHAP

Pregnancies: -0.01

ماتریس همبستگی: ضریب همبستگی Outcome Pregnancies با نشاندهنده 0.21 است که نشاندهنده رابطه مثبت ضعیف است. تأثیر این ویژگی در LIME بیشتر منفی است، در حالی که در SHAP تأثیر بسیار کمتری دارد.

نتیجه گیری نهایی :

بهطور کلی، نتایج LIME و SHAP با ماتریس همبستگی همخوانی دارند و ویژگی‌های مهم مانند BMI و Glucose تأثیر قابل توجهی در پیش‌بینی دیابت دارند در برخی ویژگی‌ها مانند SkinThickness و Insulin تفاوت‌هایی میان دو روش دیده می‌شود که نشاندهنده تفسیرهای متفاوت است

: 7

در اینجا روش Lime و Shap را برای همه نمونه‌های تست اجرا کردم . اما چون خروجی آنها حجم زیادی را داشت و برای آپلود مشکل داشتم تصمیم گرفتم خروجی را چاپ نکنم ولی کد را نوشتم . شما خوتاب برای صحت موضوع میتوانید کد را چاپ کنید و خروجی را ببینید .

بخش سوم : مدل NAM

8 الف :

تفاوت های مدل NAM با مدل Black Box هوش مصنوعی :

1. قابلیت فهم و شفافیت (Interpretability and Transparency)

مدل های black box: این مدل ها، که اغلب شامل شبکه های عصبی عمیق یا مدل های پیچیده دیگر ماشین یادگیری هستند، به دلیل ساختار پیچیده و تعداد زیاد پارامترها، معمولاً قابل فهم و تفسیر نیستند. این بدان معناست که چگونگی رسیدن به یک تصمیم یا پیش بینی برای کاربران و حتی برای توسعه دهنده ای مدل نیز نامشخص است.

مدل های NAM: این مدل ها به گونه ای طراحی شده اند که هر ویژگی ورودی به طور جداگانه توسط یک شبکه عصبی کوچک مدل سازی می شود و سپس نتایج این شبکه ها به صورت خطی ترکیب می شوند. این ساختار امکان مشاهده مستقیم تأثیر هر ویژگی بر پیش بینی نهایی را فراهم می کند، که قابلیت فهم بسیار بالایی را به ارungan می آورد.

2. پیچیدگی مدل (Model Complexity)

مدل های balck box: این مدل ها معمولاً از ساختارهای بسیار پیچیده با هزاران یا میلیون ها پارامتر استفاده می کنند. پیچیدگی بالا اغلب به دقت بیشتر منجر می شود، اما درک اینکه مدل چگونه به این نتایج دست یافته است دشوار می شود.

مدل های NAM: این مدل ها سعی می کنند تعادلی بین پیچیدگی و قابلیت فهم ایجاد کنند. هر ویژگی توسط مدل های نسبتاً ساده تری پردازش می شود که به جمع آوری و تفسیر آسان تر نتایج کمک می کند.

3. تعمیم پذیری و دقت (Generalization and Accuracy)

مدل های balck box: این مدل ها اغلب برای دستیابی به بالاترین دقت ممکن طراحی شده اند، که می تواند در برخی موارد به قیمت از دست دادن توانایی تعمیم پذیری و درک پذیری باشد.

مدل‌های NAM: اگرچه ممکن است در برخی موارد کمی کمتر دقیق باشند نسبت به برخی مدل‌های پیچیده‌تر جعبه سیاه، اما قابلیت فهم بالای آن‌ها به کاربران این امکان را می‌دهد که بفهمند چگونه پیش‌بینی‌ها تولید شده و این امر به تعمیم‌پذیری بهتر کمک می‌کند.

مزایای استفاده از مدل‌های NAM:

هر ویژگی به طور جداگانه توسط یک شبکه عصبی کوچک مدل‌سازی می‌شود، که این امر به فهم بهتر تأثیر هر ویژگی بر پیش‌بینی نهایی کمک می‌کند.

برخلاف مدل‌های جعبه سیاه، NAM‌ها فرایند تصمیم‌گیری را شفاف و قابل فهم می‌کنند، که این امر در برخی کاربردها مانند پزشکی و قضایی بسیار مهم است

مدل‌های NAM را می‌توان به آسانی برای کاربردهای مختلف تنظیم کرد و حتی به یادگیری چندوظیفه‌ای گسترش داد.

به دلیل شفافیت بالا، می‌توان بایاس‌های احتمالی در داده‌ها یا مدل را شناسایی و اصلاح کرد، که این امر در حفظ انصاف و اعتمادپذیری مدل مهم است.

معایب استفاده از مدل‌های NAM:

معمولًا دقت پایین تری نسبت به شبکه‌های عصبی عمیق مانند شبکه‌های black box دارند با وجود اینکه NAM‌ها می‌توانند توابع پیچیده‌تری نسبت به مدل‌های خطی یاد بگیرند، ممکن است در مدل‌سازی تعاملات پیچیده بین ویژگی‌ها که در برخی داده‌های واقعی وجود دارد، محدود باشند.

به دلیل استفاده از چندین شبکه عصبی کوچک، آموزش NAM‌ها ممکن است نسبت به مدل‌های ساده‌تر زمان‌برتر و مصرف منابع بیشتری داشته باشد.

برای اجرای NAM‌ها ممکن است به منابع حسابی بیشتری نیاز باشد، که این می‌تواند در محیط‌های با محدودیت منابع به یک چالش تبدیل شود

: ب

کدهای این بخش را در یک فایل جوپیتر نوت بوک به نام NAM نوشته ام.

توجه :

کد این قسمت را به صورت دستی پیاده سازی کردم. کد کتابخانه NAM را مطابق آن چیزی که در گیت هاب در کد اصلی موجود هست را به صورت دستی پیاده سازی کردم و از کتابخانه آماده NAM استفاده نکردم.

همچنین NAMClassifier را مطابق آن چیزی که در مقاله هست پیاده سازی کردم

```
class NAMClassifier(nn.Module):
    def __init__(self, num_inputs: int, num_units: List[int], hidden_sizes: List[int], dropout: float,
feature_dropout: float, activation: str = 'relu'):
        super(NAMClassifier, self).__init__()
        assert len(num_units) == num_inputs
        self.num_inputs = num_inputs
        self.num_units = num_units
        self.hidden_sizes = hidden_sizes
        self.dropout = dropout
        self.feature_dropout = feature_dropout
        self.activation = activation

        self.dropout_layer =
nn.Dropout(p=self.feature_dropout)
        self.feature_nns = nn.ModuleList([
            FeatureNN(input_shape=1,
num_units=self.num_units[i], dropout=self.dropout,
hidden_sizes=self.hidden_sizes, activation=self.activation)
            for i in range(num_inputs)
        ])
        self._bias =
torch.nn.Parameter(data=torch.zeros(1))

    def calc_outputs(self, inputs):
```

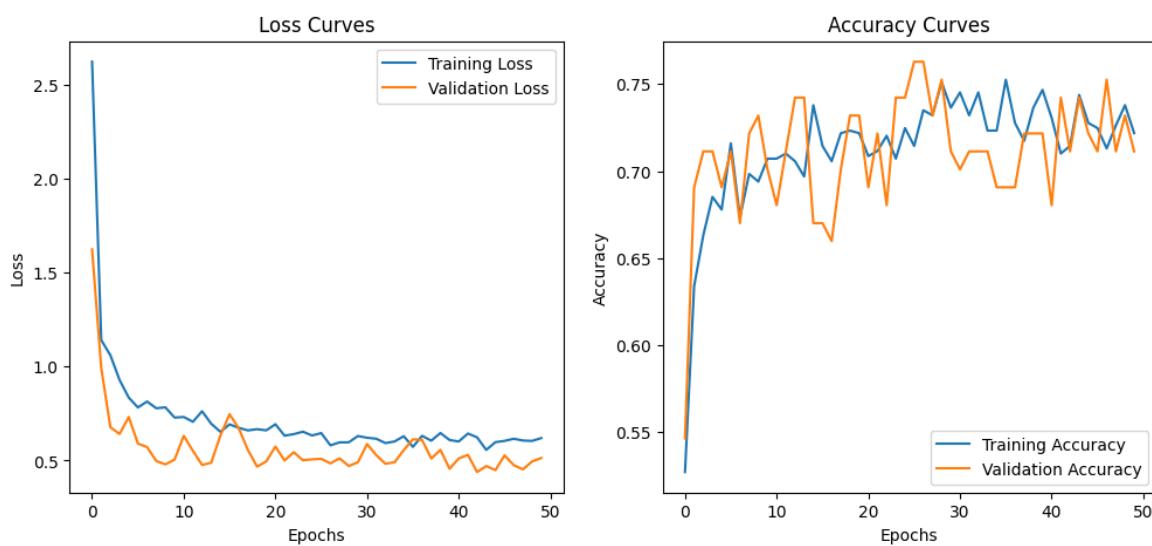
```

        return [self.feature_nns[i](inputs[:, i]) for i in
range(self.num_inputs)]

def forward(self, inputs):
    individual_outputs = self.calc_outputs(inputs)
    conc_out = torch.cat(individual_outputs, dim=-1)
    dropout_out =
self.dropout_layer(conc_out).unsqueeze(1)
    out = torch.sum(dropout_out, dim=-1)
    return out + self._bias

```

در نهایت مدل را در 50 ایپاک آموزش داده ام که نمودار تغییرات خطای دقت برای داده های آموزش و ارزیابی بدین شکل شد.



همچنین مدل را روی داده های تست ارزیابی کردم که نتایج زیر حاصل شد

معیار	مقدار
(Accuracy) دقت	0.8112
Precision	0.8182
Recall	0.8411
F1 (F1 Score) امتیاز	0.8295

Confusion Matrix

[[69 20]
[17 90]]

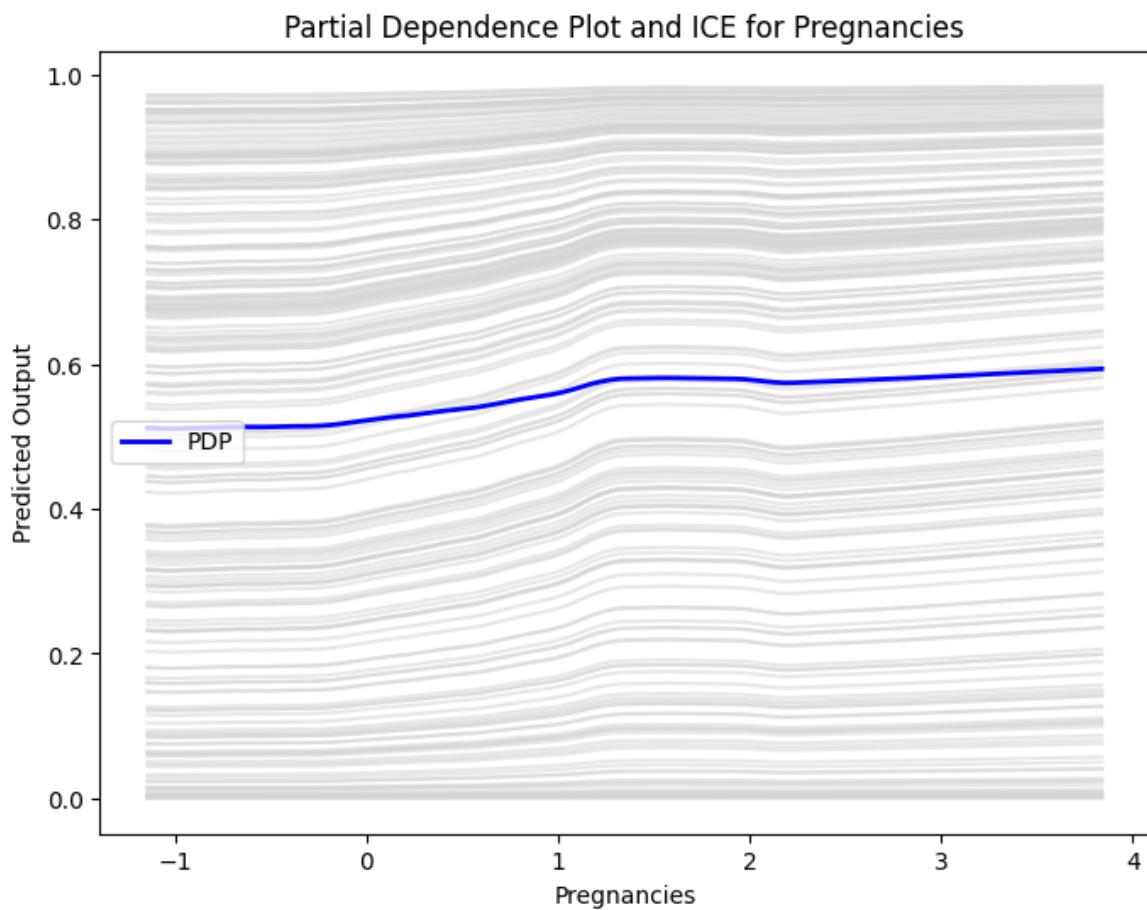
: ج

بله این روش به خوبی تفسیر پذیری مدل را بهبود بخشیده . در ادامه نتایج و تحلیل ها را می آورم .

برای هر کدام از نمودار هایی که در ادامه آورده ام دو بخش اصلی داریم :

خطوط خاکستری که هر کدام نمایانگر پیش‌بینی Individual Conditional Expectation (ICE) مدل برای یک نمونه داده با تغییر مقدار یک ویژگی هستند.

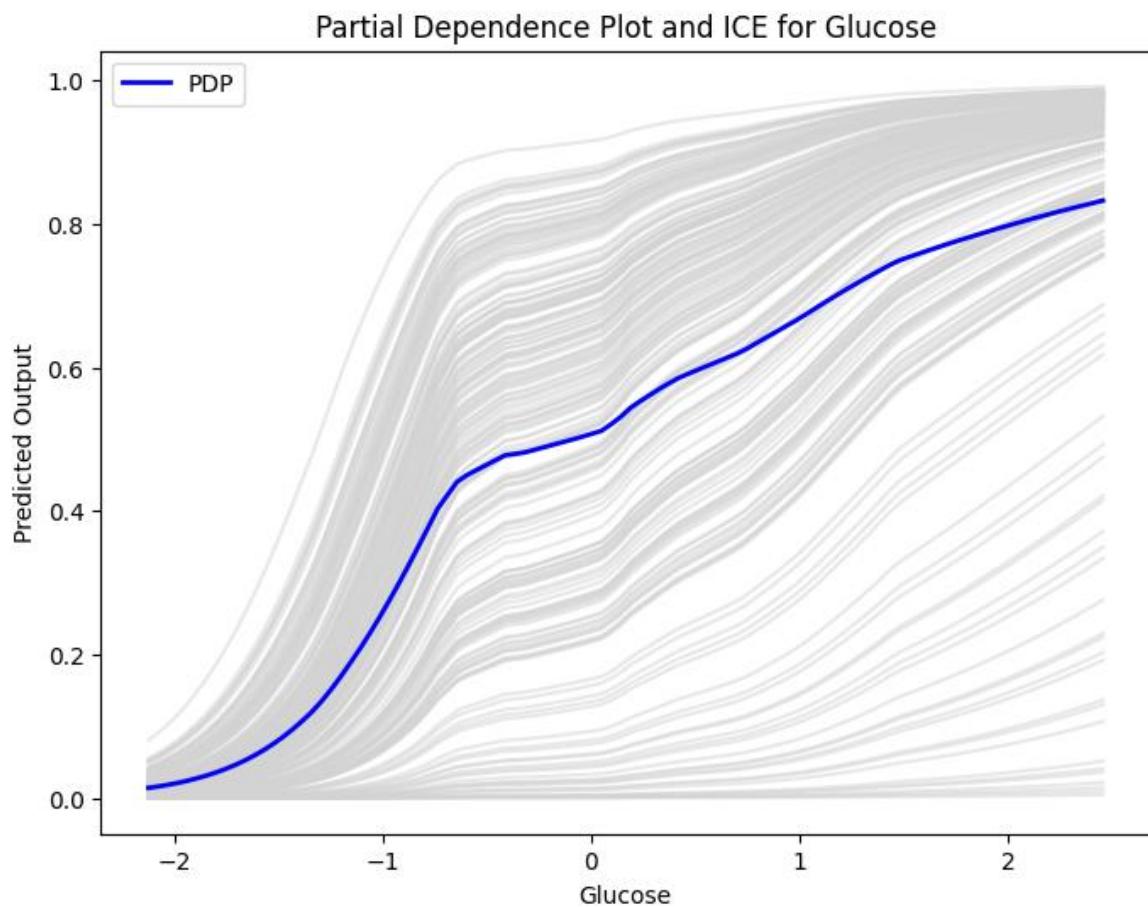
خط آبی که نمایانگر میانگین پیش‌بینی‌های مدل در طول Partial Dependence Plot (PDP) مقادیر مختلف ویژگی است.



تحلیل ICE: خطوط خاکستری نشان می‌دهند که تأثیر تعداد بارداری‌ها (Pregnancies) بر پیش‌بینی مدل در نمونه‌های مختلف داده‌ها تفاوت دارد. برخی نمونه‌ها با افزایش تعداد بارداری‌ها (Pregnancies) پیش‌بینی دیابت بالاتری دارند، در حالی که برخی دیگر تغییر چندانی ندارند. این تفاوت‌ها نشان می‌دهد که ویژگی‌های دیگر نیز نقش مهمی در پیش‌بینی مدل ایفا می‌کنند.

تحلیل PDP: خط آبی نشان‌دهنده میانگین تأثیر تعداد بارداری‌ها (Pregnancies) بر پیش‌بینی مدل است. این خط نشان می‌دهد که با افزایش تعداد بارداری‌ها، احتمال پیش‌بینی دیابت به طور کلی کمی افزایش می‌یابد. این افزایش تأثیر اندکی دارد و خط PDP تقریباً مسطح است که نشان‌دهنده تأثیر نسبی کم (Pregnancies) بر پیش‌بینی دیابت است.

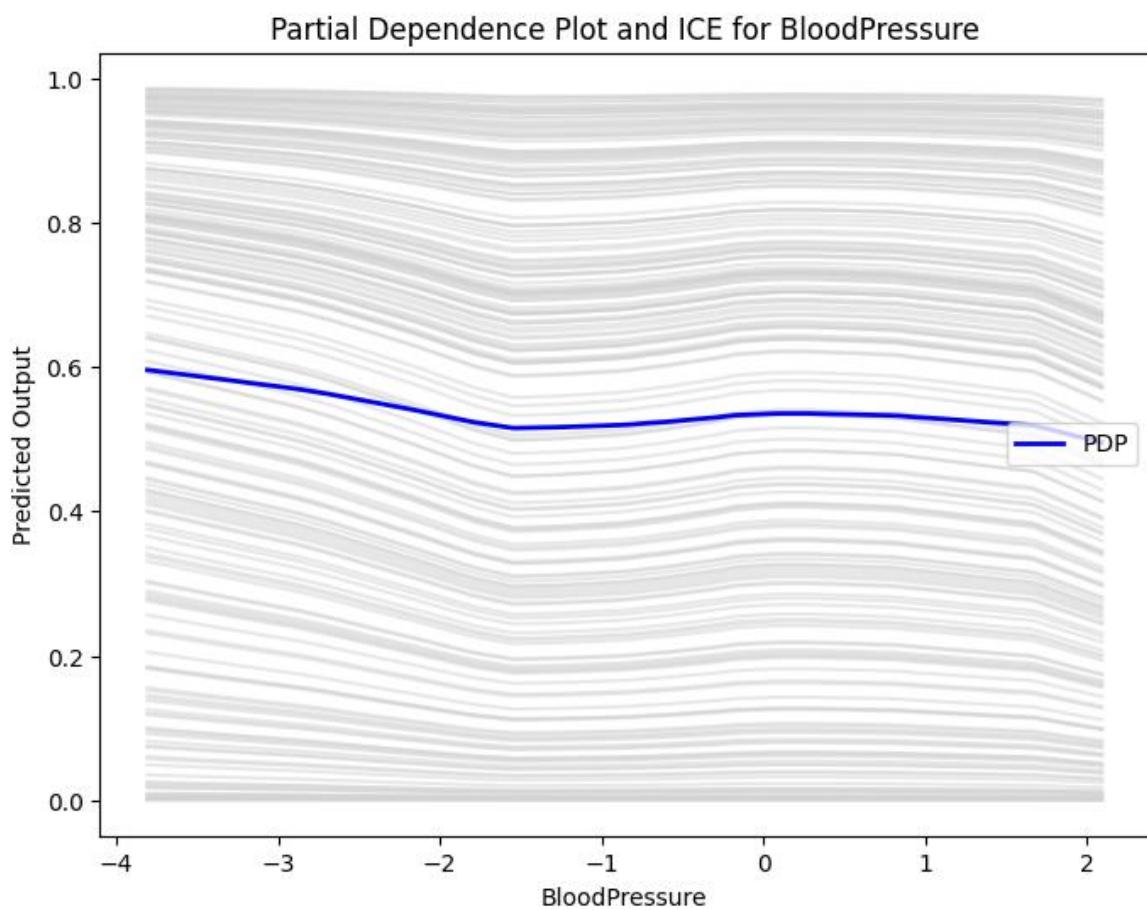
در مجموع، نمودارهای ICE و PDP هر دو نشان می‌دهند که تأثیر تعداد بارداری‌ها بر پیش‌بینی دیابت کم و مثبت است. این نتیجه با ماتریس همبستگی و تحلیل‌های دیگر همخوانی دارد.



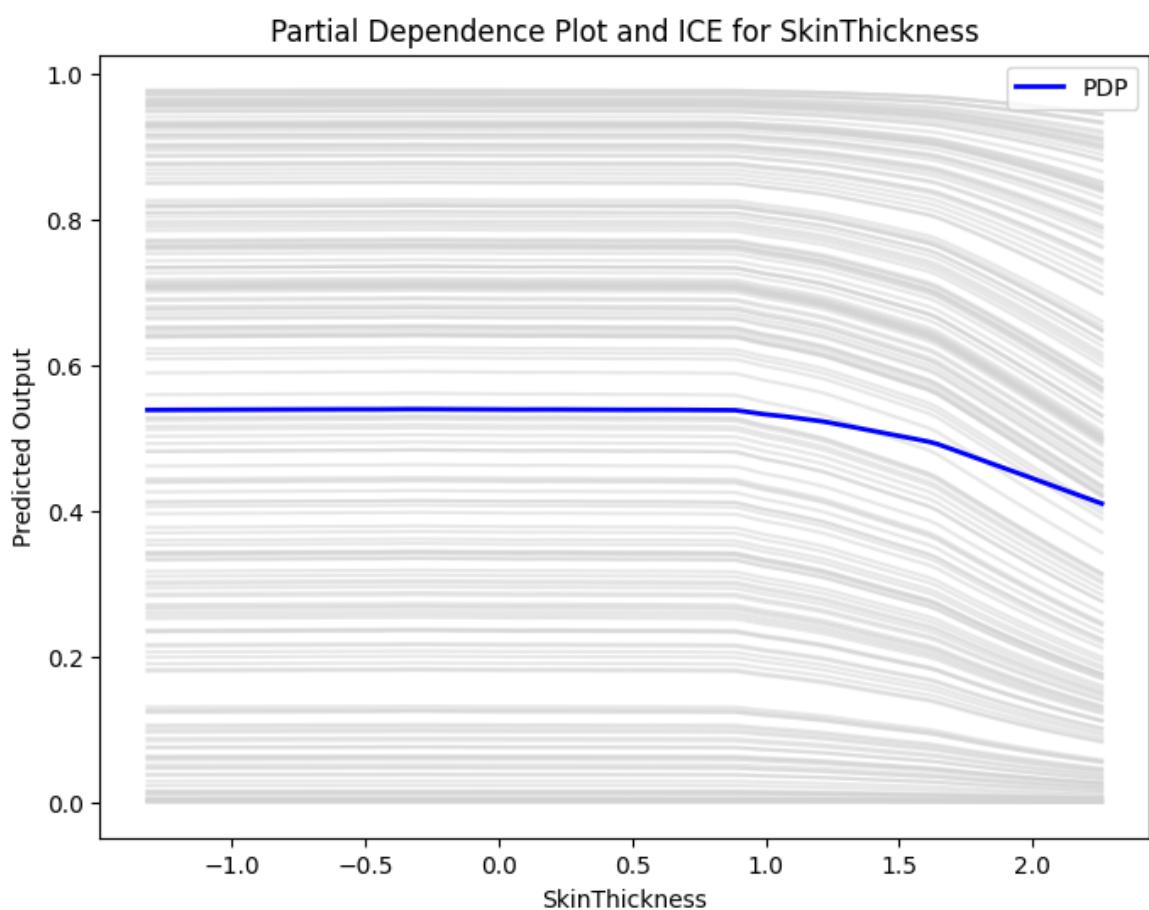
تحلیل ICE: خطوط خاکستری نشان می‌دهند که با افزایش مقدار Glucose، پیش‌بینی مدل برای اکثر نمونه‌ها افزایش می‌یابد. این نشان‌دهنده تأثیر مثبت و قوی Glucose بر پیش‌بینی دیابت است. در بعضی نمونه‌ها تأثیر افزایش Glucose بیشتر و در برخی کمتر است، اما روند کلی افزایشی است.

تحلیل PDP: خط آبی نشان‌دهنده میانگین تأثیر Glucose بر پیش‌بینی مدل است. این خط نشان می‌دهد که با افزایش مقدار Glucose، احتمال پیش‌بینی دیابت به طور قابل توجهی افزایش می‌یابد. خط PDP صعودی و تقریباً خطی است، که نشان‌دهنده تأثیر قوی و مثبت Glucose بر پیش‌بینی دیابت است.

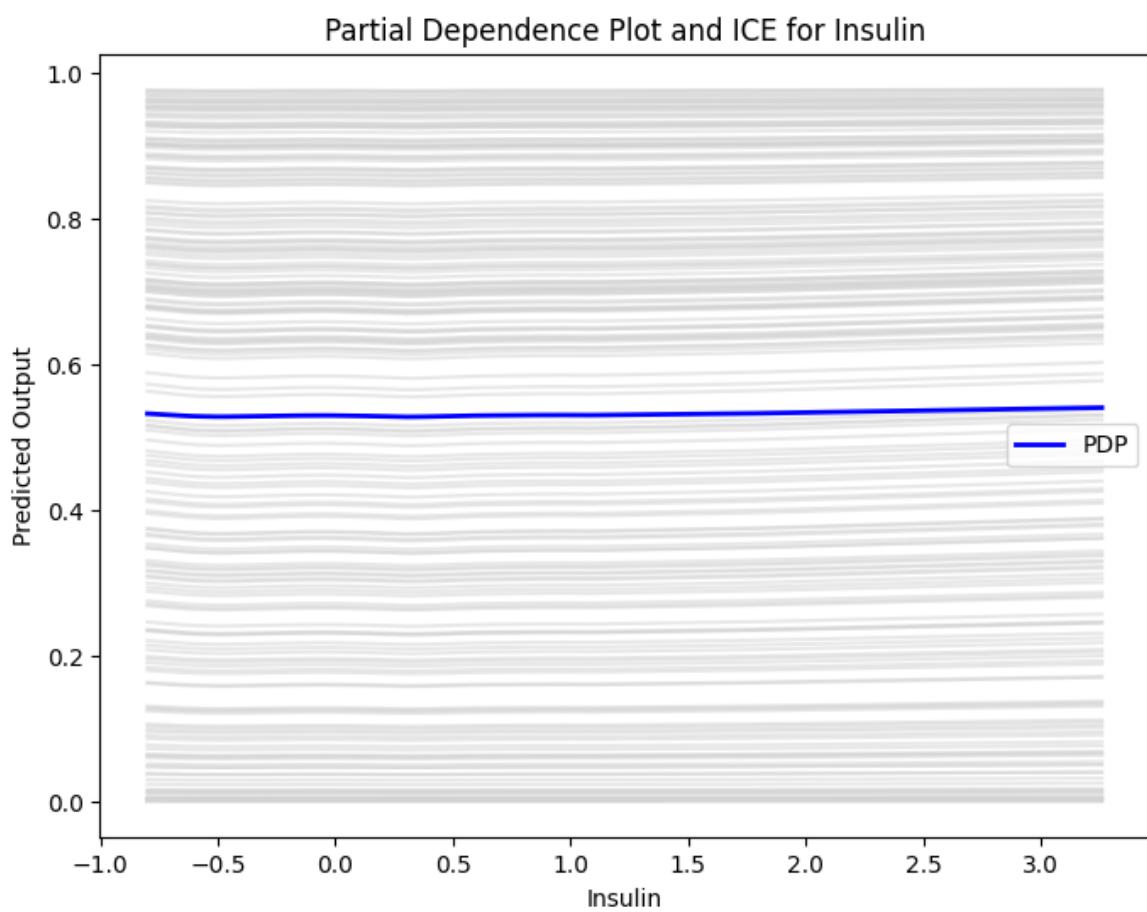
هر دو نمودار PDP و ICE نشان می‌دهند که تأثیر قوی و مثبتی بر پیش‌بینی دیابت دارد. افزایش Glucose به طور قابل توجهی احتمال دیابت را افزایش می‌دهد.



هر دو نمودار PDP و ICE نشان می‌دهند که BloodPressure تأثیر کمی بر پیش‌بینی دیابت دارد. تغییرات فشار خون به طور قابل توجهی بر احتمال دیابت تأثیر نمی‌گذارد.



هر دو نمودار ICE و PDP نشان می‌دهند که تأثیر کمی بر پیش‌بینی دیابت دارد. تغییرات ضخامت پوست به‌طور قابل توجهی بر احتمال دیابت تأثیر نمی‌گذارد



هر دو نمودار PDP و ICE نشان می‌دهند که Insulin تأثیر کمی بر پیش‌بینی دیابت دارد. تغییرات مقدار Insulin به‌طور قابل توجهی بر احتمال دیابت تأثیر نمی‌گذارد.

پاسخ ۲ - تفسیر پذیری در حوزه تصویر

GRAD-CAM-1

۱ - ۱

: ایده

Grad-CAM از اطلاعات گرادیان‌هایی که از لایه‌های آخر شبکه به سمت ورودی‌ها منتقل می‌شود برای تعیین اهمیت هر پیکسل در تصویر نسبت به کلاس مورد نظر استفاده می‌کند. این اهمیت از طریق نقشه‌های فعال‌سازی (activation maps) که نشان دهنده ویژگی‌های مورد توجه شبکه در تصویر هستند، بدست می‌آید.

: محاسبات

Forward Pass: تصویر از طریق شبکه به جلو فرستاده می‌شود تا نتیجه پیش‌بینی برای کلاس مورد نظر بدست آید.

محاسبه گرادیان: گرادیان امتیاز کلاس مورد نظر (قبل از لایه softmax) نسبت به نقشه ویژگی‌های هر کanal در لایه کانولوشنی آخر محاسبه می‌شود. فرمول آن به صورت زیر است:

$$\frac{\partial y^c}{\partial A^k}$$

که در آن y^c امتیاز کلاس و c نقشه ویژگی A^k کanal است.

Global Average Pooling (GAP): گرادیان‌های حاصل از قدم قبل بر روی ابعاد مکانی نقشه ویژگی میانگین‌گیری می‌شوند تا وزن‌های اهمیت نورون α_k^c برای کلاس c محاسبه شوند:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

که Z تعداد کل عناصر در نقشه ویژگی است (معمولاً عرض ضرب در ارتفاع)

تولید نقشه تأثیرگذاری: نقشه ویژگی‌ها با وزن‌های اهمیت آنها جمع‌بندی شده و با استفاده از تابع ReLU فیلتر می‌شوند تا تأثیرات مثبت نهایی بر کلاس مورد نظر حاصل شود:

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\sum_k \alpha_k^c A^k \right)$$

این نقشه نشان‌دهنده نواحی تصویر است که بیشترین تأثیر را بر تصمیم‌گیری شبکه داشته‌اند.

2 – 1

در اینجا میخواهیم روش Grad Cam روی مدل از پیش آموزش دیده VGG16 که با Image net آموزش دیده اعمال کنیم و نتیجه را روی 6 عکس Image net مشاهده کنیم

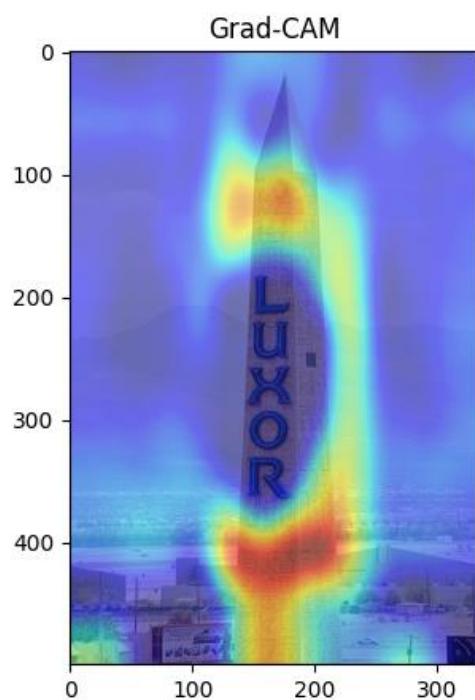
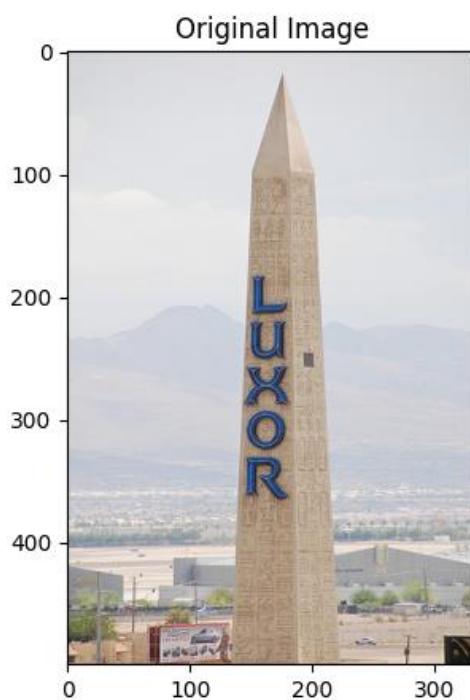
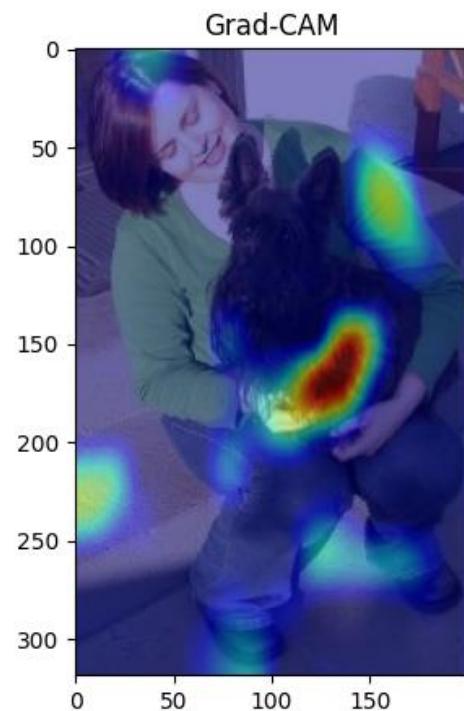
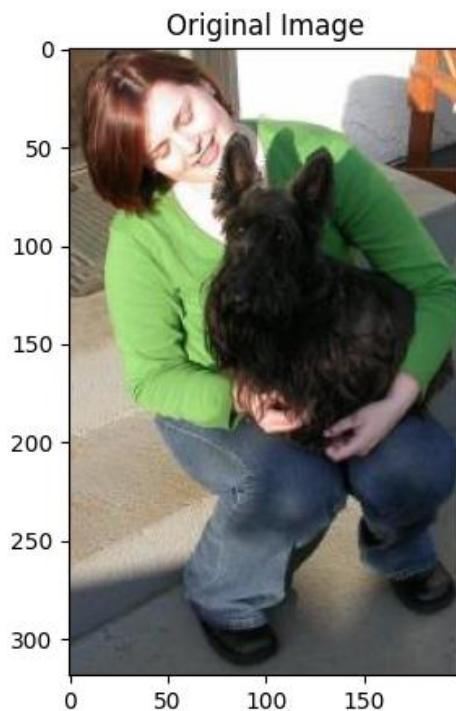
در ابتدا مدل VGG16 را بارگذاری کرده ام سپس یک تابع برای پیش پردازش نوشته ام که تصاویر ورودی را اندازه 224 در 224 تغییر اندازه می‌دهد و در نهایت آن‌ها را نرمال‌سازی می‌کند

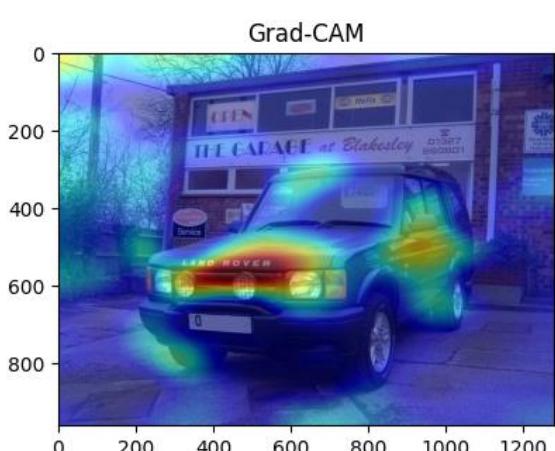
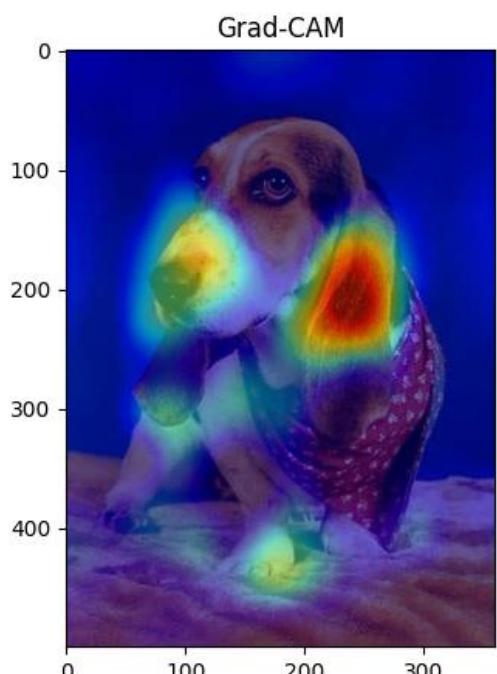
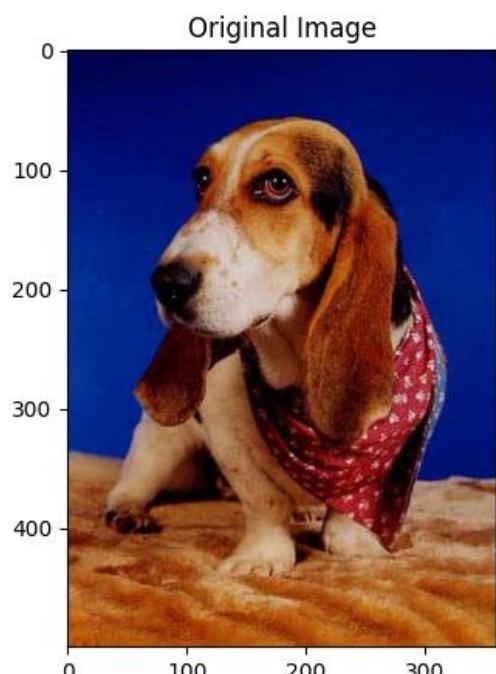
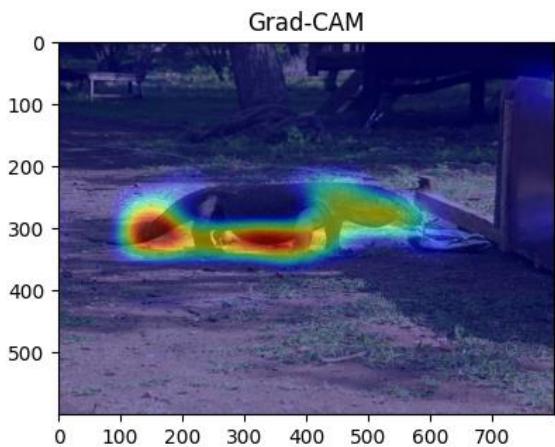
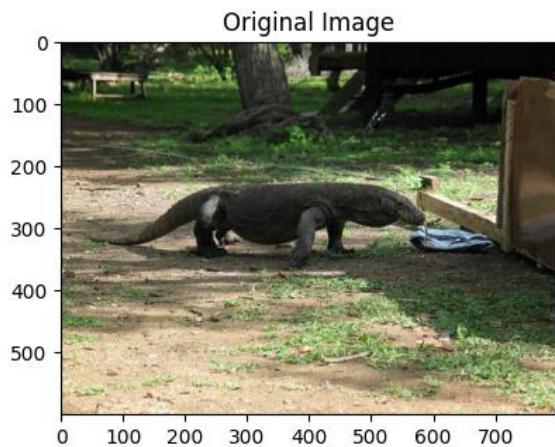
یک تابع به نام Grad-CAM نوشته ام که وظیفه اصلی آن تولید Saliency map است . در این تابع برای دریافت ویژگی‌ها و backward_hook برای دریافت گرادیان‌ها در طول عملیات‌های forward_hook رو به جلو و عقب مدل استفاده می‌شود.

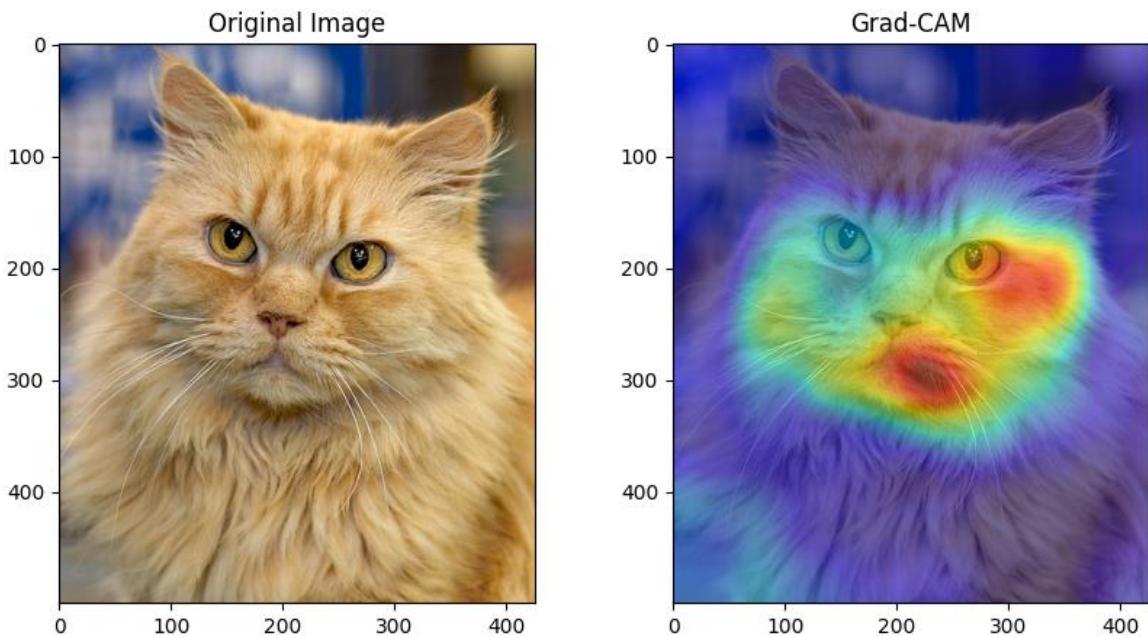
6 عکسی که از کلاس‌های متفاوت Image net استفاده کرده ام اینگونه است . (نام کلاس‌ها و شماره کلاس‌هایی که تصاویر از آنها هستند را نوشته ام)

نام کلاس	شماره کلاس
Boxer	243
Egyptian cat	285
Persian cat	281
Leonberg	208
Lynx	287
Tiger	292

در نهایت خروجی برای شش تصویر به اینگونه شد.







GUIDED GRAD-CAM

1 - 2

در واقع نوعی توسعه و تعمیم از روش محبوب Backpropagation برای آموزش شبکه‌های عصبی استفاده می‌شود. اما بر خلاف Backpropagation معمولی، این روش تمرکز بیشتری بر روی ویژگی‌هایی دارد که بیشترین تأثیر را بر پیش‌بینی خاص دارند.

در Guided Backpropagation، گرادیان‌ها از لایه خروجی به سمت لایه ورودی باز می‌گردند، اما با این تفاوت که فقط گرادیان‌های مثبت (یعنی تأثیر مثبت بر خروجی) اجازه عبور دارند. این کار باعث می‌شود تا تنها ویژگی‌هایی که مثبتاً بر نتیجه تأثیر گذاشته‌اند، نمایان شوند.

مزیت اصلی روش Guided Backpropagation نسبت به Saliency Backpropagation معمولی در ایجاد آن است که تصاویر تولید شده توسط Guided Backpropagation تمایز ویژگی‌های مهم برای تصمیم‌گیری‌های مدل را با دقت و وضوح بیشتری نشان می‌دهند.

Guided Backpropagation با فیلتر کردن و تمرکز بر گرادیان‌های مثبت، فقط اجزایی که تأثیر مثبتی بر پیش‌بینی دارند را مورد توجه قرار می‌دهد. این کار باعث می‌شود که Saliency Maps تولیدی توسط این روش تمیز و واضح‌تر باشند، و در نتیجه، تصاویر ساده‌تر و قابل فهم‌تری از دلایل پیش‌بینی‌های مدل ارائه می‌دهند.

در مقایسه با Backpropagation معمولی، که ممکن است Saliency Maps تولیدی حاوی نویز زیادی باشد، Guided Backpropagation با حذف گرادیان‌های منفی می‌تواند نویز را کاهش دهد.

2 – 2

در این قسمت مانند قسمت قبل مدل VGG16 را آپلود کردم و پیش‌پردازش‌های اولیه را مانند قبل وارد کردم تابعی به نام guided_backprop به صورت زیر نوشتیم

```
def guided_backprop(model, image, target_class):
    image = preprocess(image).unsqueeze(0)
    image.requires_grad = True

    model.zero_grad()
    output = model(image)
    target = output[0][target_class]
    target.backward()

    grad = image.grad.data[0].numpy()
    grad = np.maximum(grad, 0) # Apply ReLU to gradients

    return grad
```

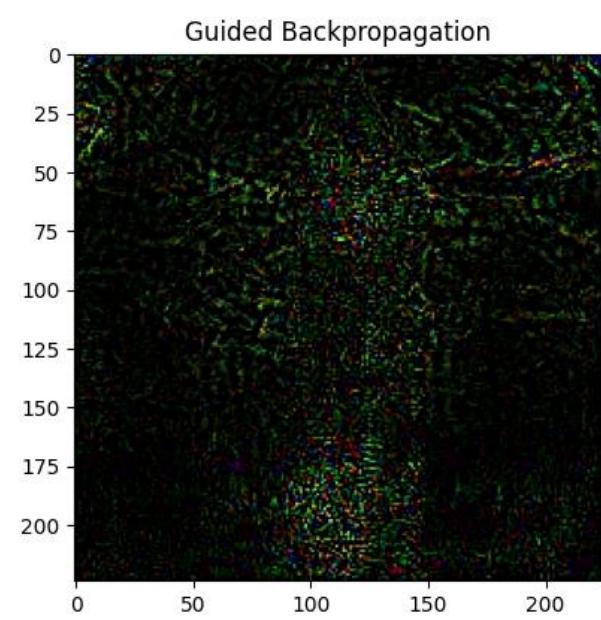
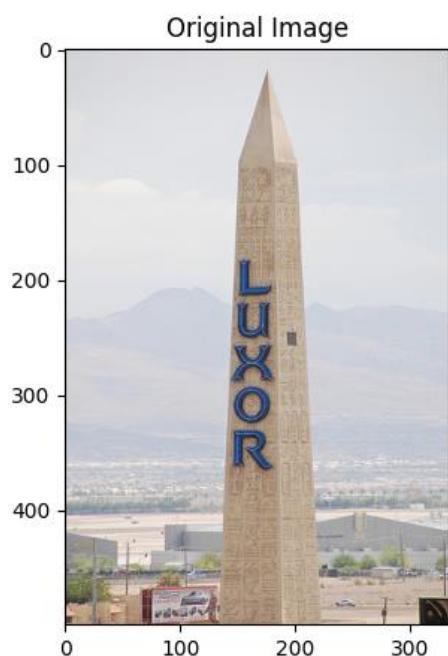
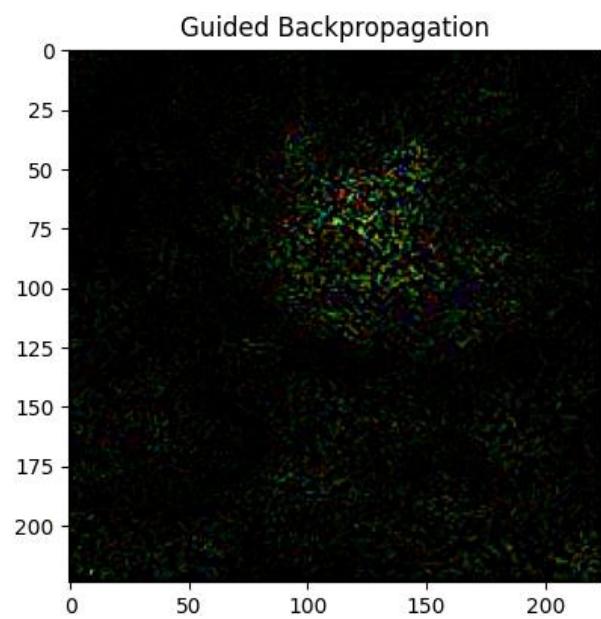
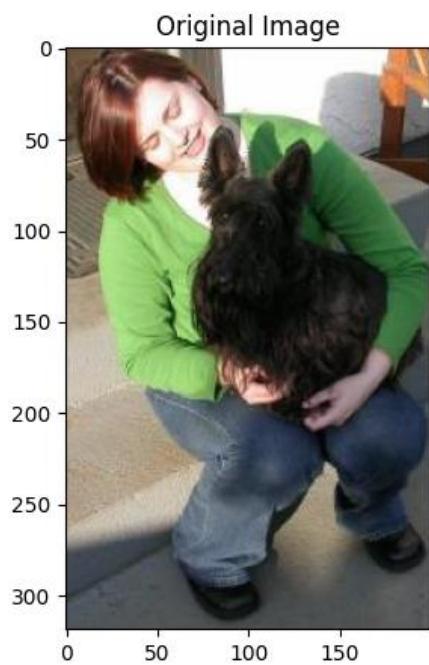
این تابع، نقشه تأثیرگذاری Guided Backpropagation را تولید می‌کند:

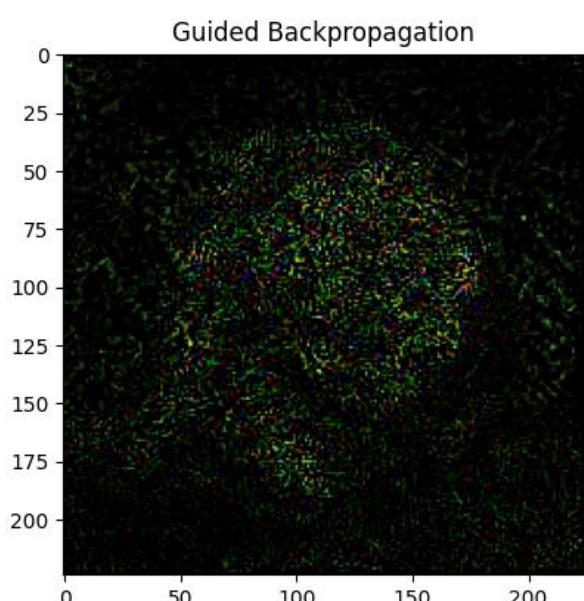
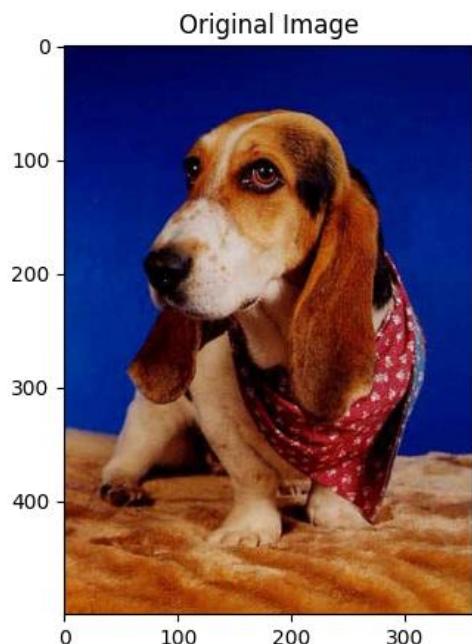
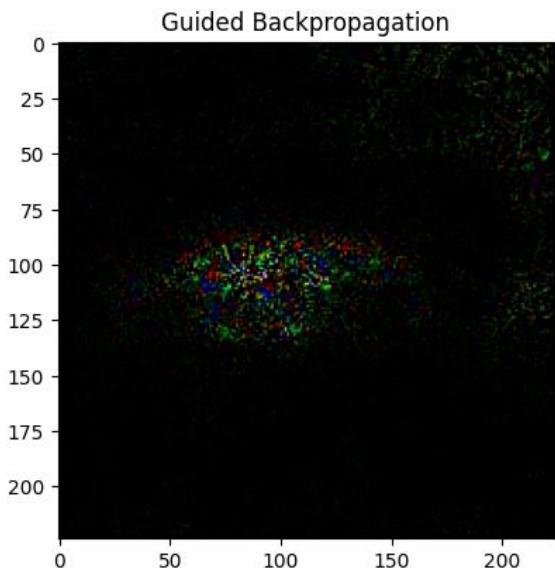
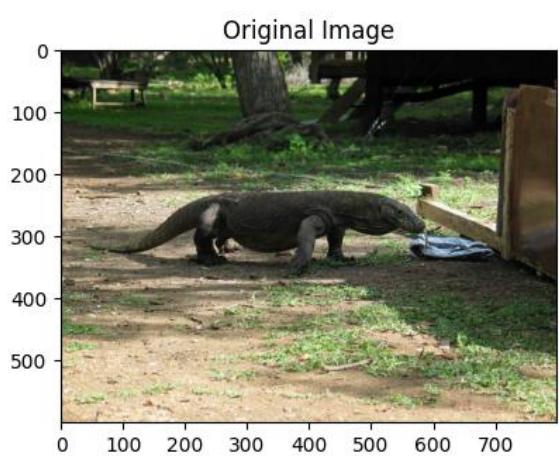
تصویر ورودی پردازش و به مدل فرستاده می‌شود و مدل یک پیش‌بینی انجام می‌دهد و گرادیان‌ها محاسبه می‌کند و در نهایت گرادیان‌ها محاسبه و فقط مقادیر مثبت حفظ می‌شوند (توسط ReLU).

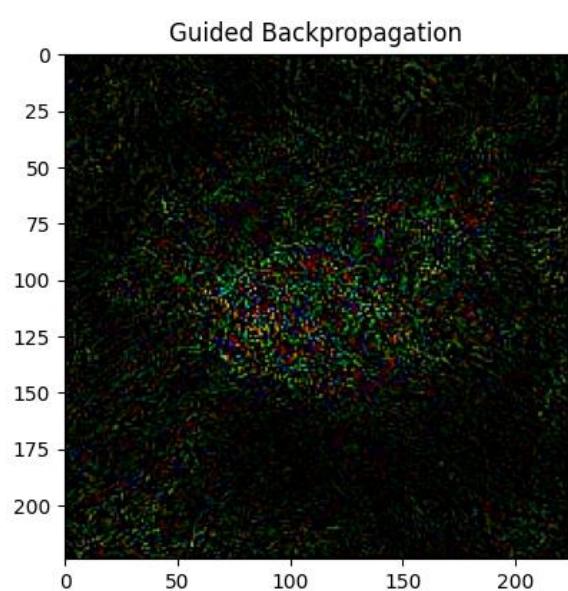
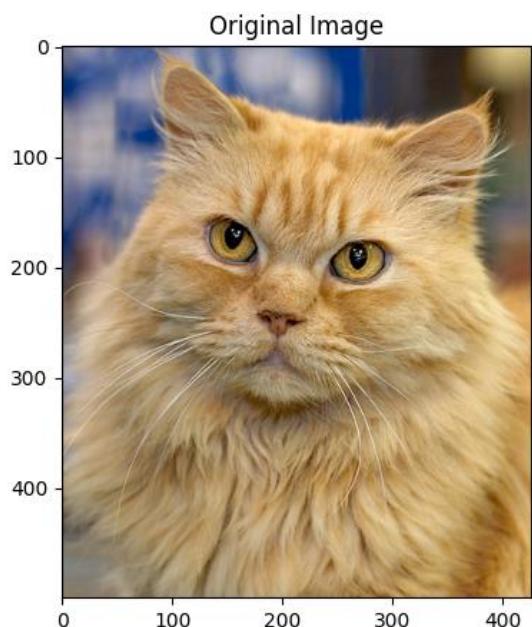
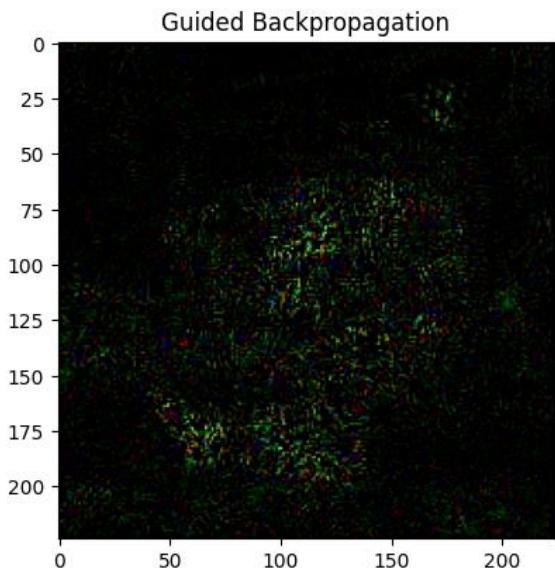
توجه : در کد به علت بهبود وضوح و نمایش بهتر از افزایش کنtras است (contrast) استفاده کردم . که این کار باعث می‌شود جزئیات مهم تصویر بهتر نمایش داده شود .

(کنtras) (Contrast) به تفاوت بین روشنایی (brightness) و تاریکی (darkness) در یک تصویر اشاره دارد. این تفاوت میزان تمایز و وضوح اجزای مختلف تصویر را تعیین می‌کند. افزایش کنtras معمولاً باعث می‌شود که تفاوت‌ها بین بخش‌های روشن و تاریک بیشتر شود و جزئیات تصویر به وضوح بیشتری دیده شوند)

نتایج این قسمت روی همان شش تصویر به این گونه شد







3 – 2

یک روش پیشرفته و ترکیبی برای تولید saliency maps است که از دو تکنیک Guided-CAM پایه یادگیری عمیق Grad-CAM و Guided Backpropagation استفاده می‌کند. هدف از این ترکیب، افزایش قدرت تفسیرپذیری مدل‌های یادگیری عمیق است.

Guided Backpropagation به تنها‌ی بر تأکید گرادیان‌های مثبت و حذف گرادیان‌های منفی تمرکز می‌کند تا تصاویری شفاف و دقیق از ویژگی‌های مهم برای تصمیم‌گیری مدل ارائه دهد. از طرف دیگر، Grad-CAM با استفاده از گرادیان‌های محاسبه شده نسبت به ویژگی‌های خروجی نقشه‌های فعال‌سازی مرتبط با کلاس خاصی را وزن‌دهی می‌کند تا نشان دهد کدام مناطق از تصویر برای تصمیم‌گیری‌های مدل حیاتی هستند.

Grad-CAM از Guided-CAM برای تعیین نقاط کلیدی یا نواحی مهم در تصاویر استفاده می‌کند که بیشترین تأثیر را بر پیش‌بینی‌های شبکه دارند. Grad-CAM با استفاده از میانگین وزن‌دار گرادیان‌ها برای لایه‌های کانولوشنی، نقشه حرارتی (heatmap) از ویژگی‌های برجسته را ایجاد می‌کند. سپس، با استفاده از Guided Backpropagation، که گرادیان‌های مثبت را در سراسر شبکه تقویت می‌کند، این نقشه‌های حرارتی به صورت واضح‌تر و تمیز‌تر تجسم می‌شوند.

هدف از ترکیب Guided Backpropagation و Grad-CAM

وضوح بالاتر: ترکیب گرادیان‌های مثبت از Guided Backpropagation با تولید Grad-CAM به تولید نقشه‌های برجستگی با وضوح بالاتر و کاهش نویز منجر می‌شود.

دقت تفسیر: این روش ترکیبی اجازه می‌دهد تا ویژگی‌های دقیق‌تر و مرتبط‌تری از تصاویر را مشاهده کنیم که بر تصمیم‌گیری‌های مدل تأثیر می‌گذارند، که این امر به تفسیر بهتر و دقیق‌تر مدل‌ها منجر می‌شود.

بهینه‌سازی مدل‌ها: با فهم عمیق‌تر از چگونگی عملکرد مدل و اینکه چه ویژگی‌هایی بیشترین تأثیر را بر پیش‌بینی‌ها دارند، می‌توان به بهینه‌سازی و تنظیم دقیق‌تر پارامترهای مدل پرداخت.

در این بخش روش Guided Grad-CAM را روی مدل پیاده سازی کردم . کدی که نوشتمن سه بخش اصلی دارد : ایجاد Grad-CAM، اجرای Guided Backpropagation، و ترکیب این دو برای تولید نقشه‌های ویژگی

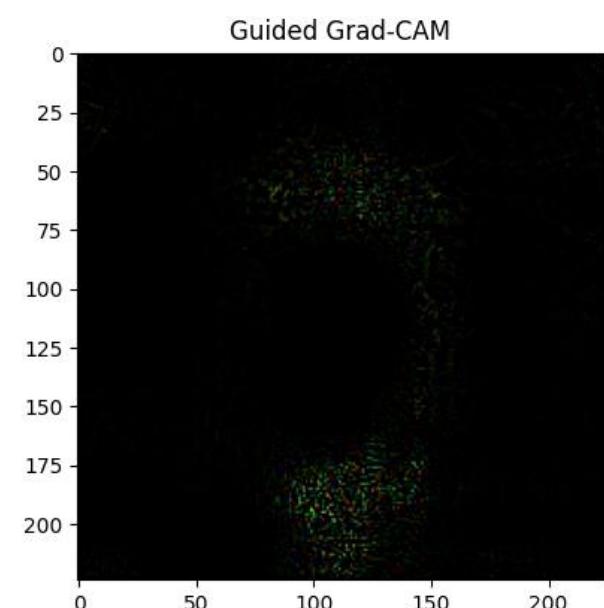
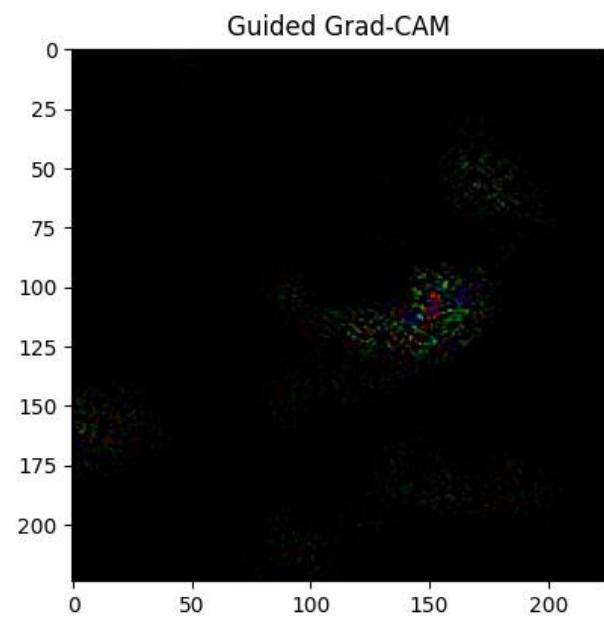
تابعی به نام guided_grad_cam نوشته ام که این تابع نتایج تولید شده توسط دو تابع قبلی را ترکیب می‌کند تا نقشه ویژگی بصری و دقیق‌تری ارائه دهد که هم تأثیر مناطق مختلف تصویر و هم تأثیر پیکسل‌های خاص را در بر دارد.

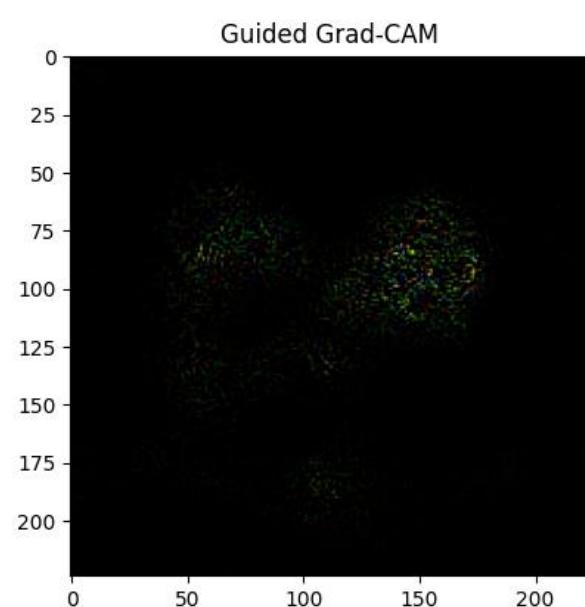
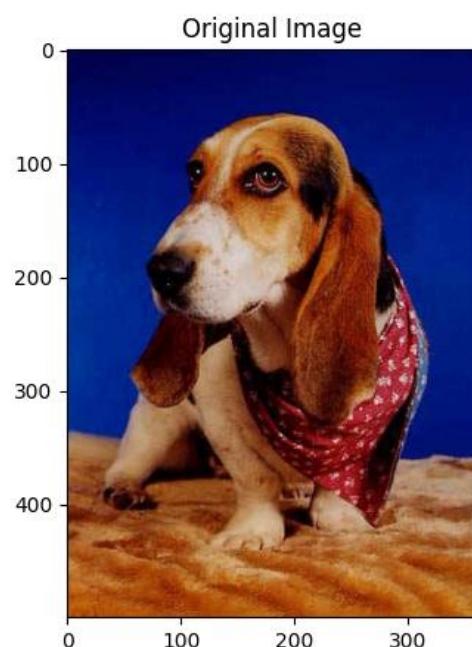
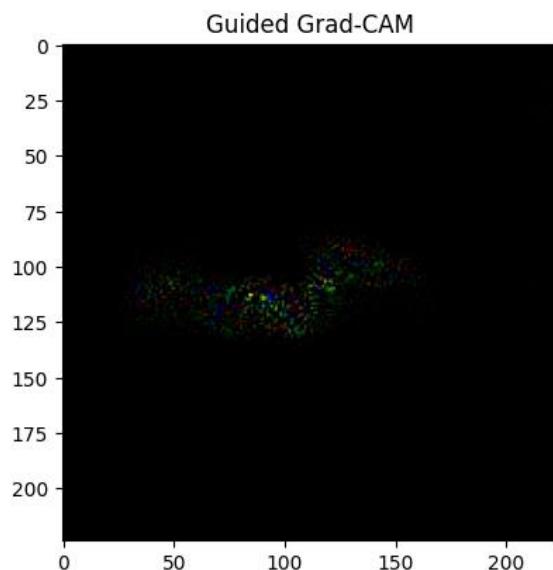
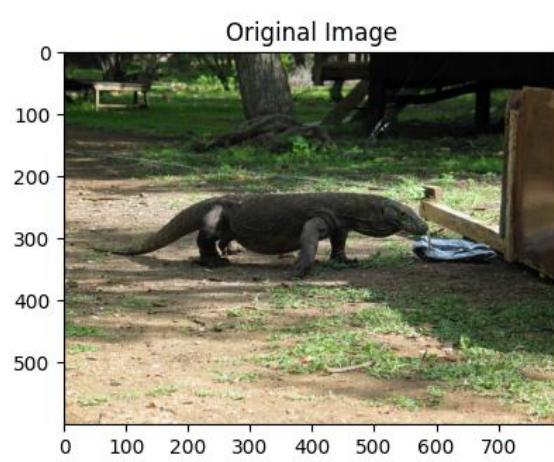
```
def guided_grad_cam(model, image, target_layer, target_class):
    cam = grad_cam(model, image, target_layer, target_class)
    guided_grad = guided_backprop(model, image, target_class)

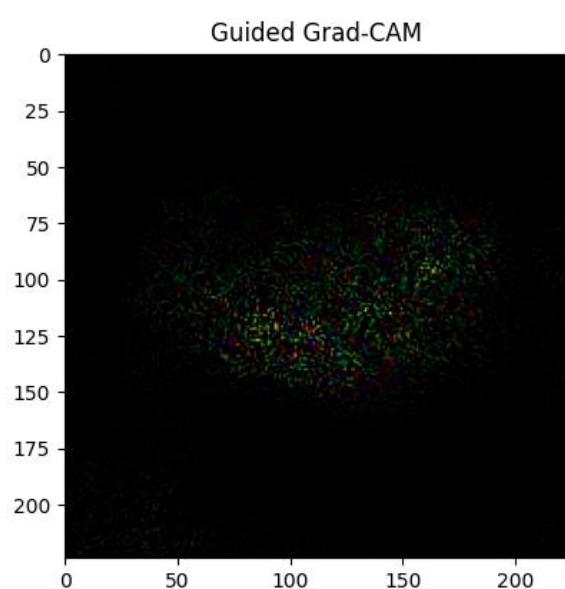
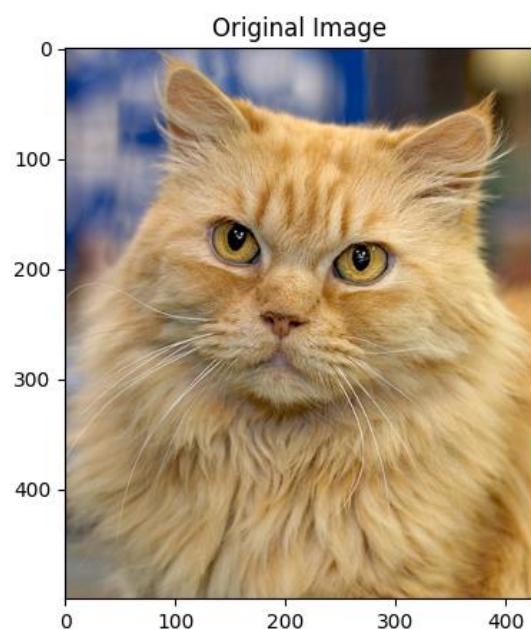
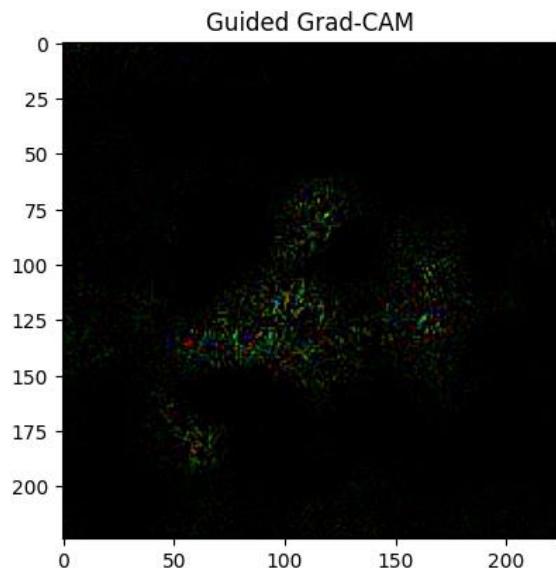
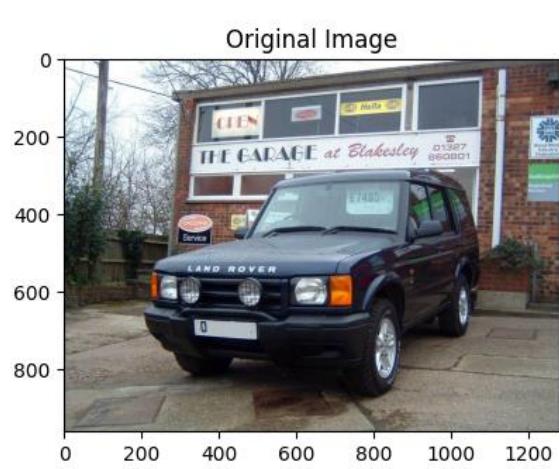
    # Ensure the dimensions match
    cam = cam / 255.0
    cam = np.expand_dims(cam, axis=0) # Add a channel dimension
    guided_grad = guided_grad * cam

    return guided_grad
```

در نهایت خروجی Saliency map تصاویر به این گونه شد .







SMOOTHGRAD

1 - 3

روش SmoothGrad برای کاهش نویز بصری در نقشه‌های حساسیت، که از طریق گرادیان‌های مدل‌های یادگیری عمیق محاسبه می‌شوند، طراحی شده است. این روش به صورت خاص برای بهبود قابلیت تفسیرپذیری Saliency map‌ها در شبکه‌های عصبی کانولوشنی (CNN) به کار می‌رود. در زیر گام‌های اصلی که SmoothGrad دنبال می‌کند، را توضیح داده ام :

افزودن نویز تصادفی: به تصویر ورودی چندین بار نویز گوسی اضافه می‌شود. این نویز به صورت تصادفی برای هر نمونه تولید می‌شود، که موجب ایجاد نمونه‌های مختلفی از تصویر اصلی با کمی تغییرات جزئی می‌شود.

محاسبه نقشه‌های حساسیت برای هر نمونه: برای هر نمونه تصویری که با نویز تغییر یافته، نقشه حساسیت محاسبه می‌شود. این نقشه‌ها بر اساس گرادیان خروجی شبکه نسبت به ورودی تصویر محاسبه می‌شوند.

میانگین‌گیری نقشه‌های حساسیت: نقشه‌های حساسیت تولید شده از تمامی نمونه‌ها میانگین گرفته می‌شود. این میانگین‌گیری به کاهش نویزهای تصادفی و غیرمرتبط کمک می‌کند و نقشه حساسیت نهایی را واضح‌تر و دقیق‌تر می‌سازد.

مقاله سعی داشته وضوح نقشه‌های حساسیت را بهتر کند و نویز آن‌ها هم کاهش دهد. نقشه‌های حساسیت سنتی، که اغلب بر پایه گرادیان‌های خام هستند، می‌توانند شامل نویز بصری زیادی باشند که موجب سردرگمی تفسیر می‌شوند. SmoothGrad با میانگین‌گیری گرادیان‌ها از نمونه‌های متعدد با نویز، نویز را کاهش داده و نقشه‌هایی شفاف‌تر و معتبرتر از لحاظ بصری ارائه می‌دهد، که بهتر می‌توانند نقاط مهم و تأثیرگذار تصویر را برجسته کنند.

در این قسمت میخواهیم هر دو روش SmoothGrad و Guided Backpropagation را روی تصاویر اعمال کنیم

ابتدا نحوه کار هر روش را به صورت خلاصه یادآوری میکنم.

:Guided Backpropagation

این روش از گرادیان‌های مثبت استفاده می‌کند تا نواحی مهم تصویر را که بیشترین تأثیر را بر تصمیم نهایی مدل دارند، برجسته کند.

در تابع `guided_backprop`, تصویر به یک `Tensor` تبدیل می‌شود و سپس از مدل عبور می‌کند تا گرادیان‌ها محاسبه شوند.

گرادیان‌های منفی به صفر تنظیم می‌شوند تا فقط گرادیان‌های مثبت باقی بمانند

:SmoothGrad

این روش با اضافه کردن نویز به تصویر اصلی و میانگین‌گیری نقشه‌های حساسیت حاصل از تصاویر نویزدار، نویز موجود در نقشه‌های حساسیت را کاهش می‌دهد.

در تابع `smooth_grad`, نویز گوسی به تصویر اصلی اضافه می‌شود و برای هر تصویر نویزدار، نقشه حساسیت با استفاده از روش Guided Backpropagation محاسبه می‌شود.

نقشه‌های حساسیت میانگین‌گیری می‌شوند تا نقشه حساسیت نهایی حاصل شود که نویز کمتری دارد.

برای این قسمت یک تابع به نام `smooth_grad` به صورت زیر نوشته ام.

```

# Function for SmoothGrad
def smooth_grad(model, image, target_class, n_samples=50, noise_level=0.2):
    image_np = np.array(image).astype(np.float32) / 255.0
    image_np = preprocess(image).unsqueeze(0).numpy()
    smooth_grad = np.zeros_like(image_np)

    for i in range(n_samples):
        noisy_image_np = image_np + np.random.normal(0, noise_level, image_np.shape)
        noisy_image_np = np.clip(noisy_image_np, 0, 1)
        noisy_image = torch.tensor(noisy_image_np).float()
        noisy_image = noisy_image.squeeze(0)
        noisy_image_pil = transforms.ToPILImage()(noisy_image)
        grad = guided_backprop(model, noisy_image_pil, target_class)
        smooth_grad += grad

    smooth_grad /= n_samples
    return smooth_grad

```

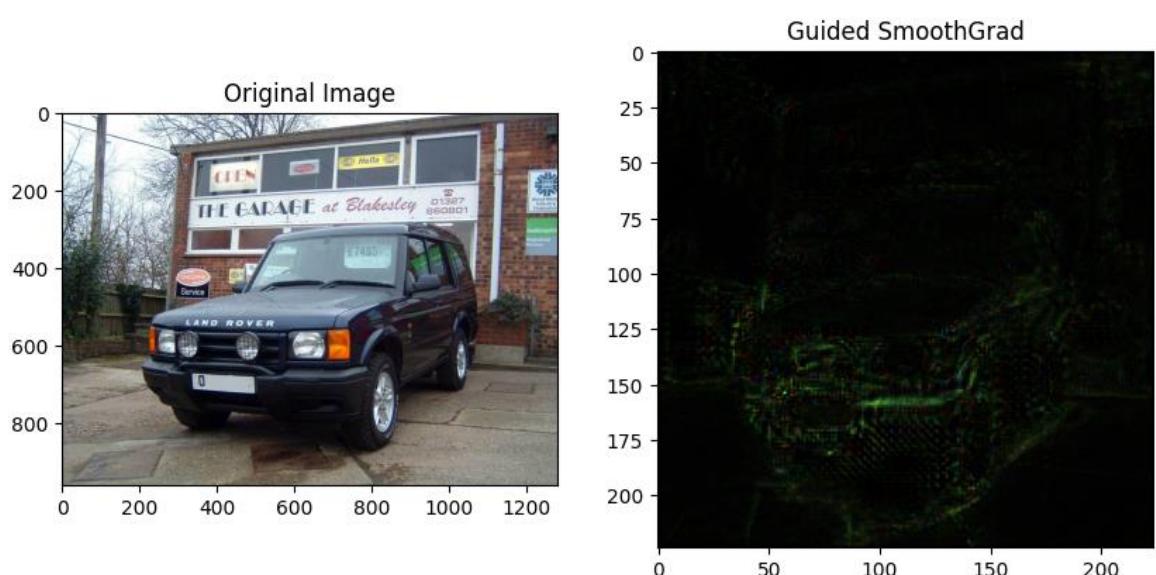
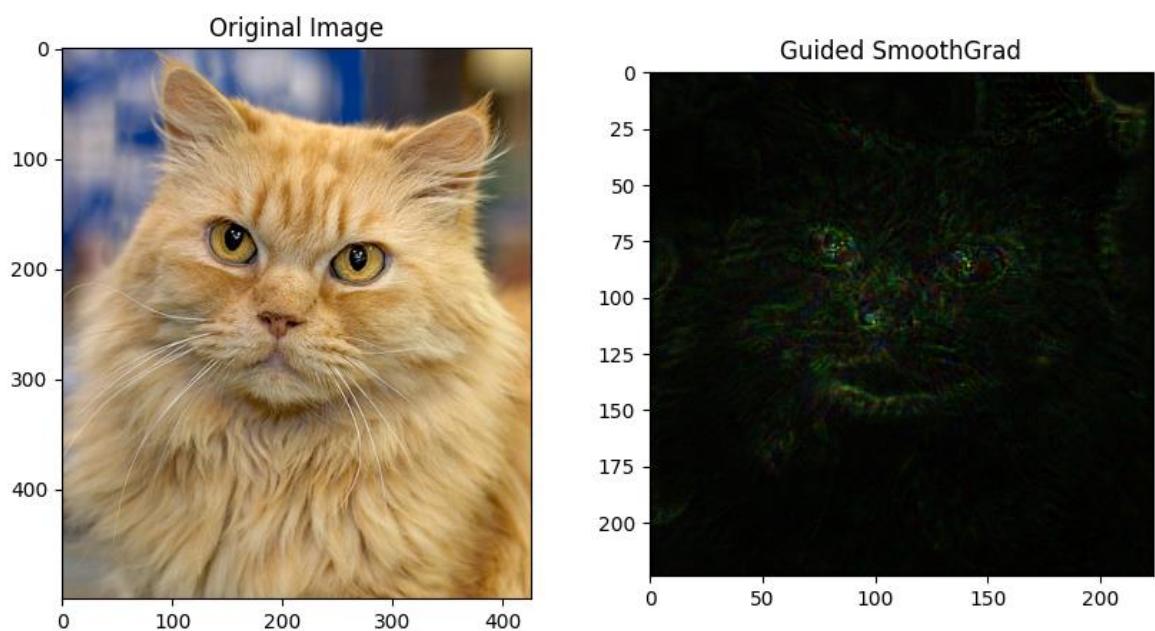
این تابع تصویر ورودی پردازش می‌شود و نویز گوسی به آن اضافه می‌شود. برای هر تصویر نویزدار، نقشه حساسیت با استفاده از روش Guided Backpropagation محاسبه می‌شود. نقشه‌های حساسیت میانگین‌گیری می‌شوند تا نقشه حساسیت نهایی حاصل شود.

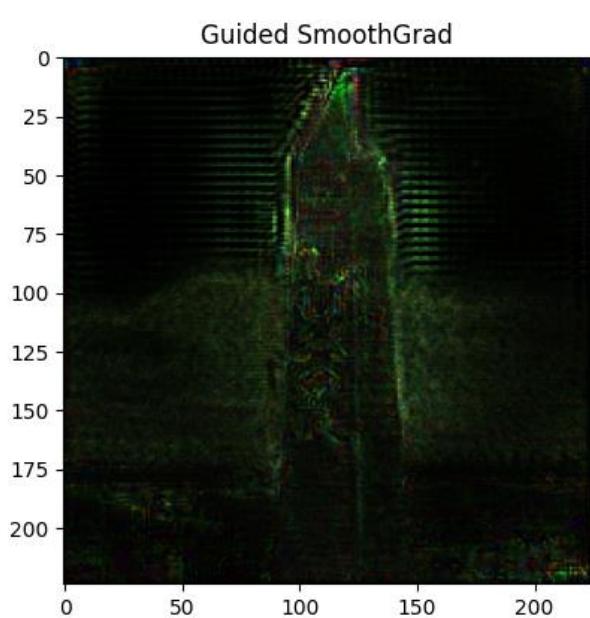
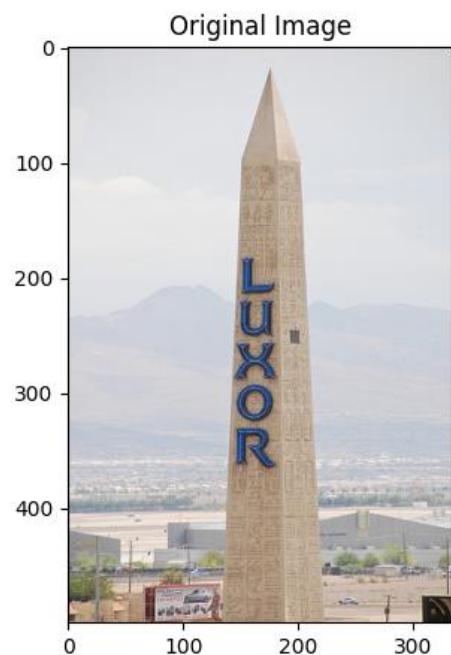
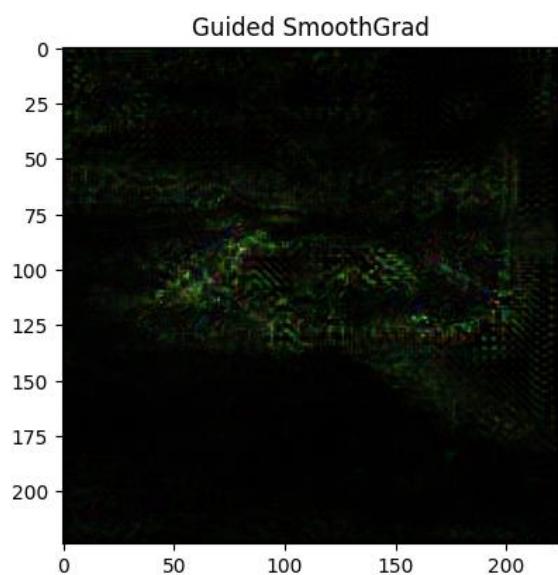
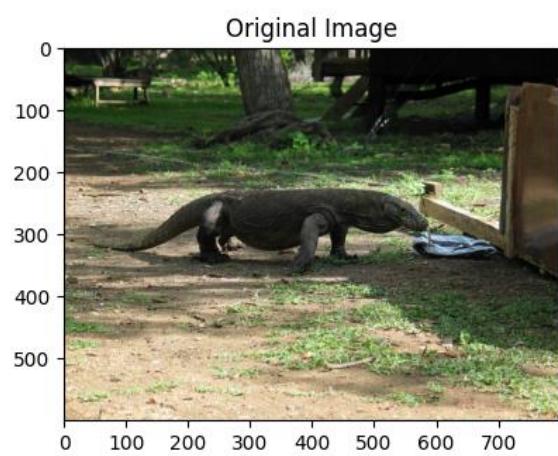
برای پیدا کردن مقادیر تعداد نمونه‌ها (n_samples) و سطح نویز (noise_level)، من حدس و خطاهای متعددی را زدم و در نهایت به مقدارهای زیر رسیدم که به نظر مقادیر مناسبی باشند.

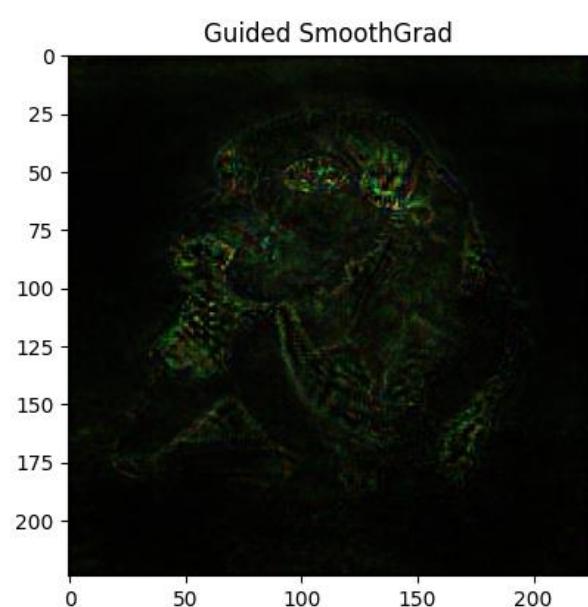
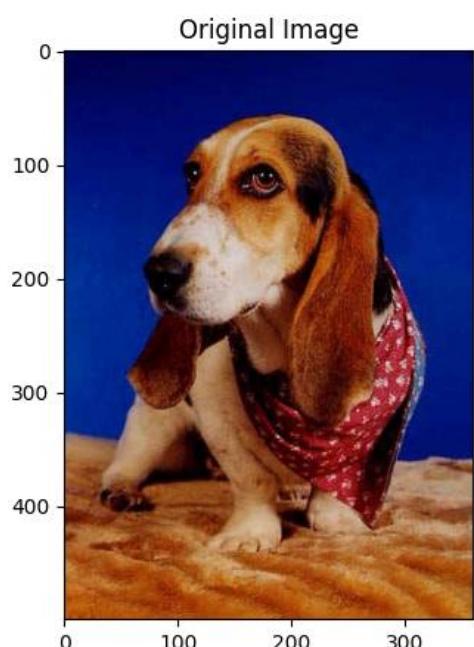
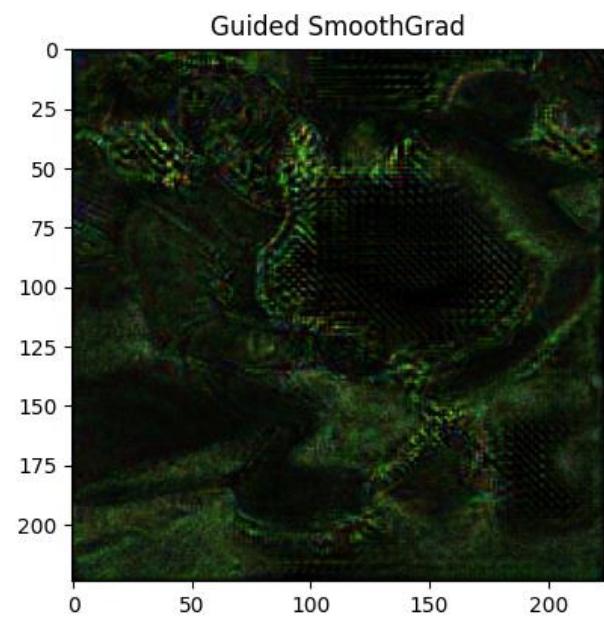
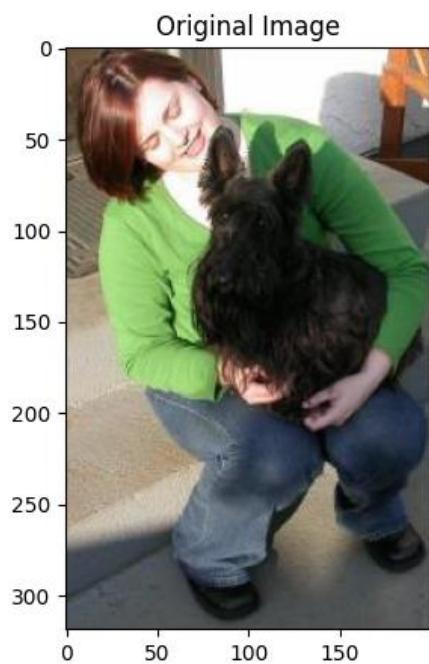
تعداد نمونه‌ها : 50

سطح نویز : 0.2

نتایج زیر از این روش حاصل شد :







نتیجه: Saliency map بیشتر تصاویر به خوبی تولید شد به جز یک تصویر بقیه تصاویر نتایج بسیار مناسبی داشتند.

3 -3

کدهای این قسمت را در یک فایل جوپیتر نوت بوک به نام

SmoothGrad+Guided_Backpropagation+Grad_CAM قرار داده ام

در این بخش میخواهم روشی که در بخش قبل پیاده سازی کردم (Guided_SmoothGrad)

را با Grad-CAM ادغام کنم و نتیجه را مشاهده کنیم.

برای این کار یک تابع به نام Combined Grad-CAM به صورت زیر نوشته ام.

```
# Function to combine SmoothGrad + Guided Backpropagation with Grad-CAM
def combined_grad_cam(model, image, target_layer, target_class, n_samples=50, noise_level=0.2):
    grad_cam_map = grad_cam(model, image, target_layer, target_class)
    smooth_grad_map = smooth_grad(model, image, target_class, n_samples, noise_level)

    # Normalize the smooth_grad_map
    smooth_grad_map = smooth_grad_map[0] # Remove the batch dimension
    smooth_grad_map = np.transpose(smooth_grad_map, (1, 2, 0))
    smooth_grad_map = smooth_grad_map - smooth_grad_map.min()
    smooth_grad_map = smooth_grad_map / smooth_grad_map.max()

    # Resize smooth_grad_map to match grad_cam_map
    smooth_grad_map_resized = np.array(Image.fromarray((smooth_grad_map * 255).astype(np.uint8)).resize(grad_cam_map.shape[:-1], Image.ANTIALIAS))

    # Convert grad_cam_map to range [0, 1]
    grad_cam_map = grad_cam_map / 255.0

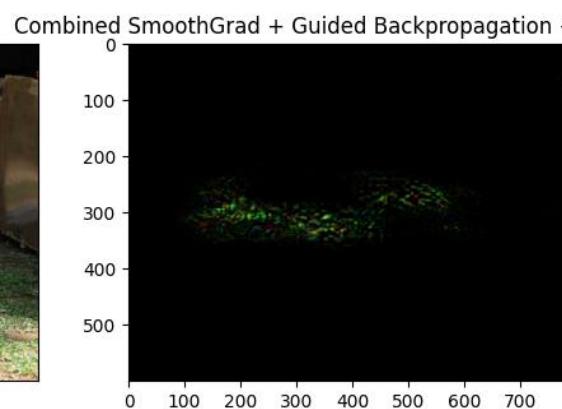
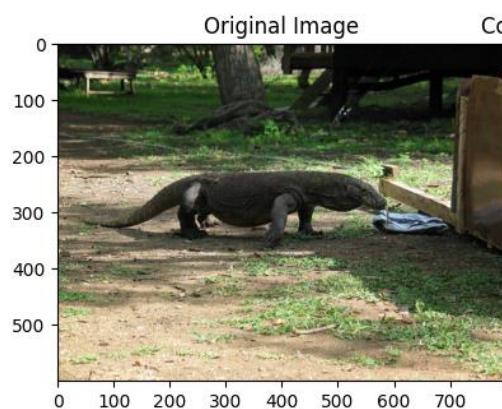
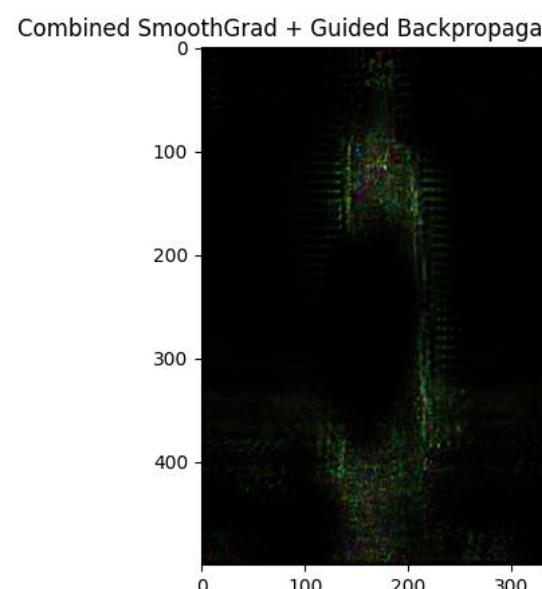
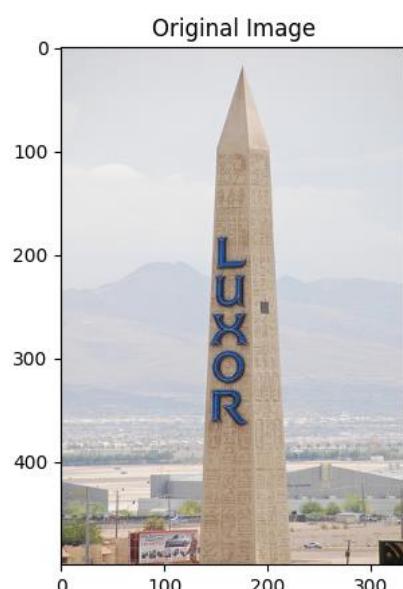
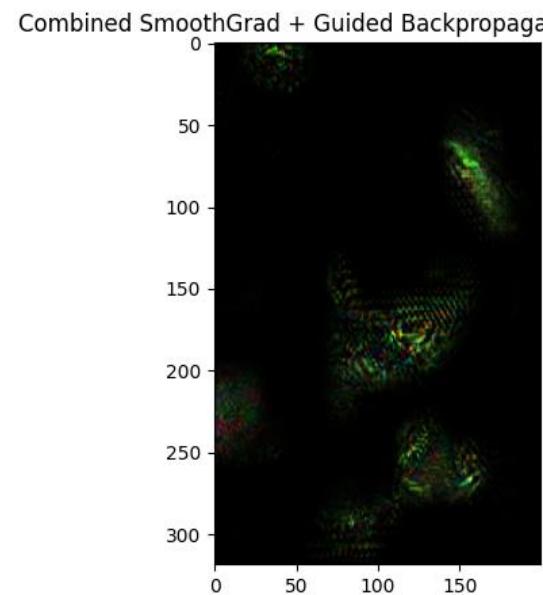
    # Combine the Grad-CAM and SmoothGrad maps
    combined_map = grad_cam_map[..., np.newaxis] * smooth_grad_map_resized # Broadcasting
    combined_map = combined_map - combined_map.min()
    combined_map = combined_map / combined_map.max()

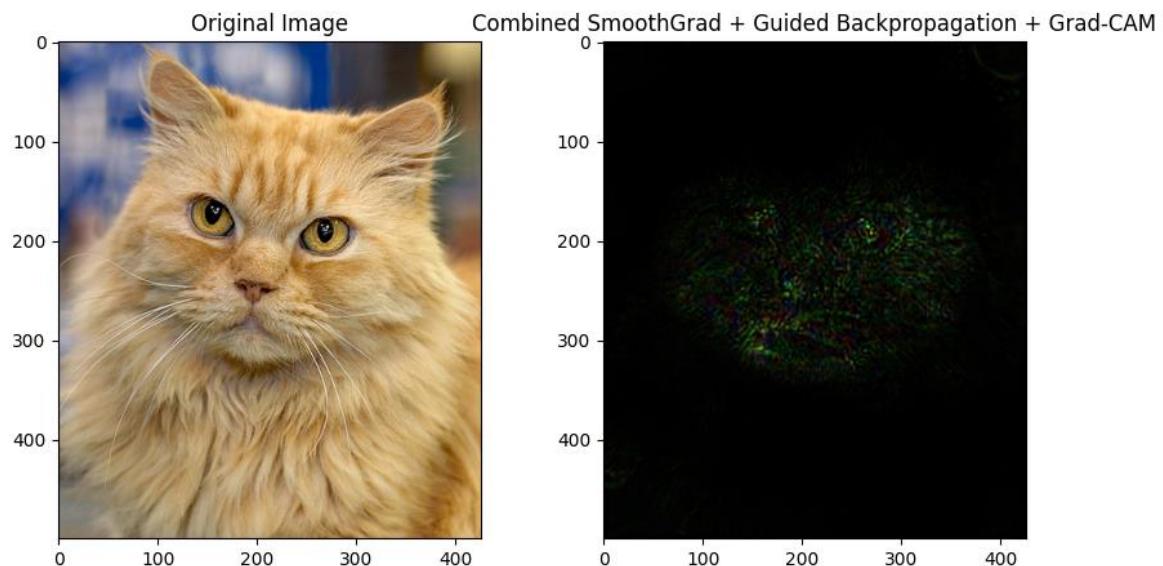
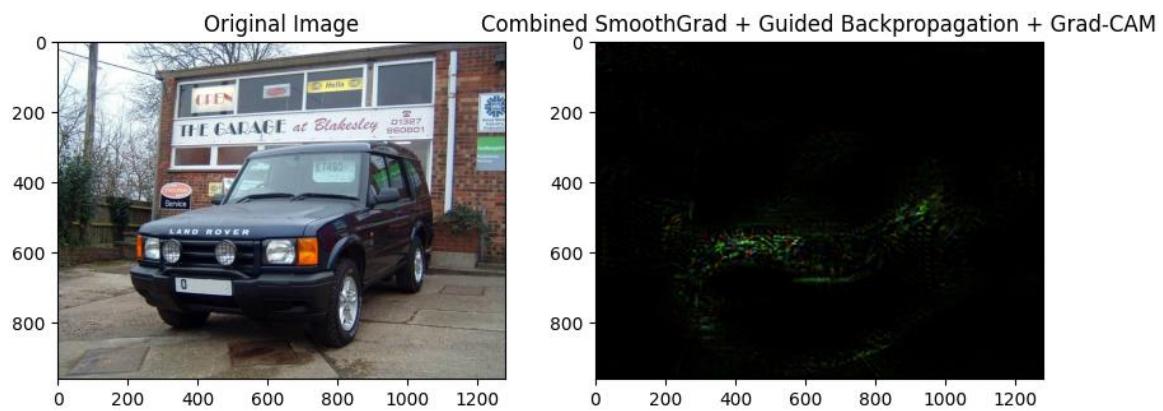
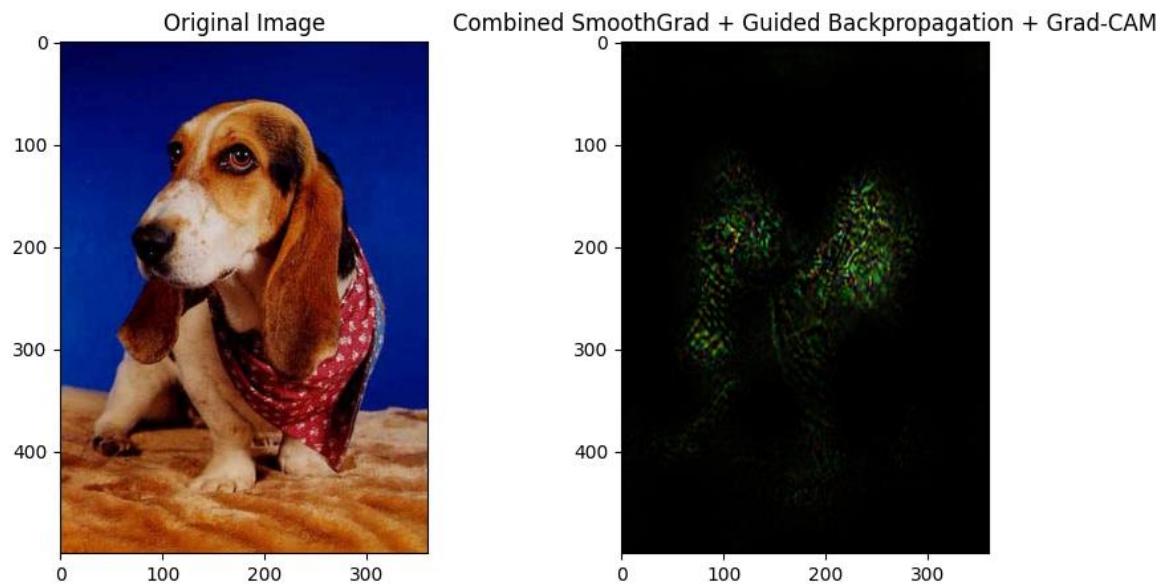
    return combined_map
```

تابع بالا Guided_SmoothGrad و Grad-CAM را تولید شده توسط Saliency map می‌کند تا تصویری دقیق‌تر و واضح‌تر از نواحی مهم و تاثیرگذار تصویر ارائه دهد. این ترکیب به درک بهتر نحوه تصمیم‌گیری شبکه کمک می‌کند

مقدار هایپر پارامتر ها را همان هایپر پارامتر های قبلی یعنی تعداد نمونه 50 و سطح نویز 0.2 قرار داده ام.

خروجی این روش به این گونه شد.





ADVERSARIAL PETREBUTION AND PIXEL ATTRIBUTION

1-4

کد های این قسمت را در همان فایل SmoothGrad+Guided_Backpropagation+Grad_CAM نوشته ام

برای این قسمت میخواهیم یک نمونه Adversarial از یکی از عکس های داده شده بدست آوریم و سپس آن را با روش قبل بدست آوریم.

برای این کار من از نمونه Persian Cat استفاده کردم و با روش FGSM نمونه Adversarial را ایجاد کرده ام. کلاس اصلی عکس 283 هست ولی کلاس نمونه Adversarial آن تغییر میکند و 281 میشود.

Original Class: 283, Predicted Class after Attack: 281

در کد داده شده من یک تابع به نام fgsm_attack به صورت زیر نوشته ام.

```
# Function for FGSM attack
def fgsm_attack(image, epsilon, data_grad):
    sign_data_grad = data_grad.sign()
    perturbed_image = image + epsilon * sign_data_grad
    perturbed_image = torch.clamp(perturbed_image, 0, 1)
    return perturbed_image
```

این تابع، تصویر ورودی، میزان اغتشاش (epsilon) و گرادیان تصویر را گرفته و با افزودن گرادیان به تصویر، تصویر مختل شده را ایجاد می کند. این تصویر برای ایجاد تغییرات مخرب در خروجی مدل به کار می رود.

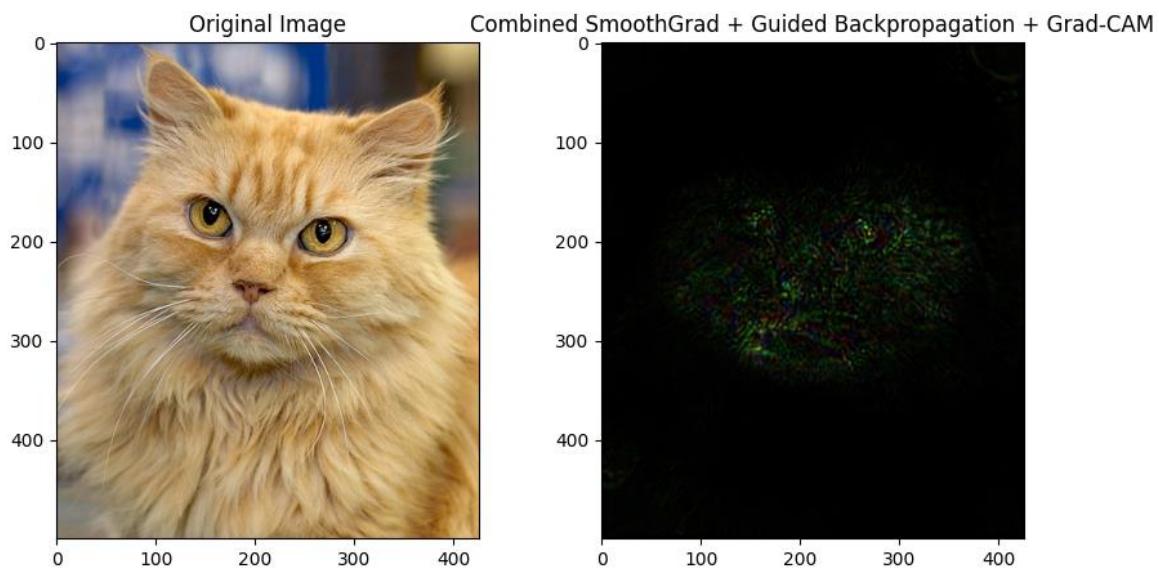
همچنین تابع دیگری به نام generate_perturbed_image نوشته ام این تابع از مدل، تصویر، کلاس هدف و میزان اغتشاش برای ایجاد یک تصویر مختل شده استفاده می کند. این تابع از مدل، برای محاسبه خروجی و گرادیان استفاده کرده و سپس تابع fgsm_attack را فراخوانی می کند.

```

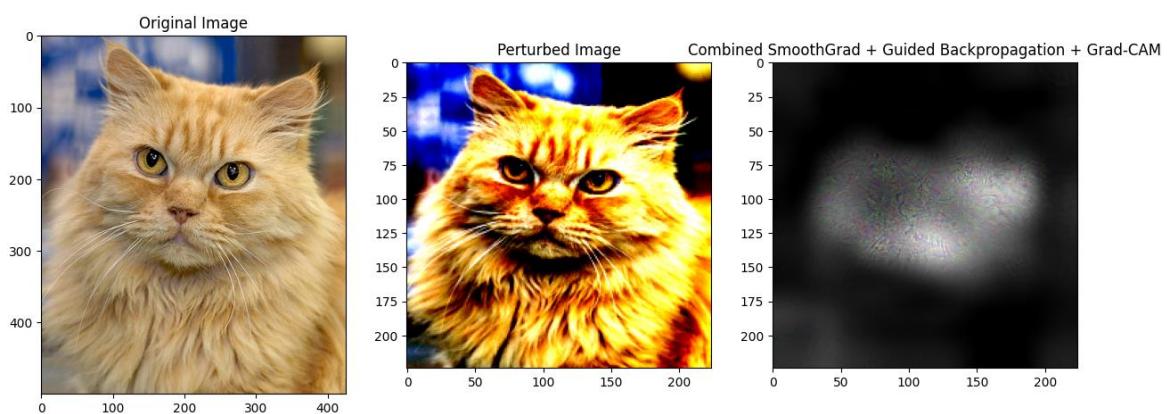
# Function to generate perturbed image
def generate_perturbed_image(model, image, target_class, epsilon=0.01):
    image.requires_grad = True
    output = model(image)
    loss = F.nll_loss(F.log_softmax(output, dim=1), torch.tensor([target_class]))
    model.zero_grad()
    loss.backward()
    data_grad = image.grad.data
    perturbed_image = fgsm_attack(image, epsilon, data_grad)
    return perturbed_image

```

خروجی Adversarial تصویر اصلی و نمونه Saliency map آن به این گونه شد.



تصویر اصلی و Saliency map آن



نمونه Adversarial تصویر و Saliency map آن

مشاهده میشود که در نمونه Adversarial مدل کاملا به جاهای دیگری از تصویر توجه کرده است و این باعث شده است که مدل به اشتباہ بیافتد و لیبل تصویر را اشتباہ تشخیص دهد . و Saliency map چیزی کاملا متفاوت با تصویر اصلی شده است .

FEATURE VISUALIZATION

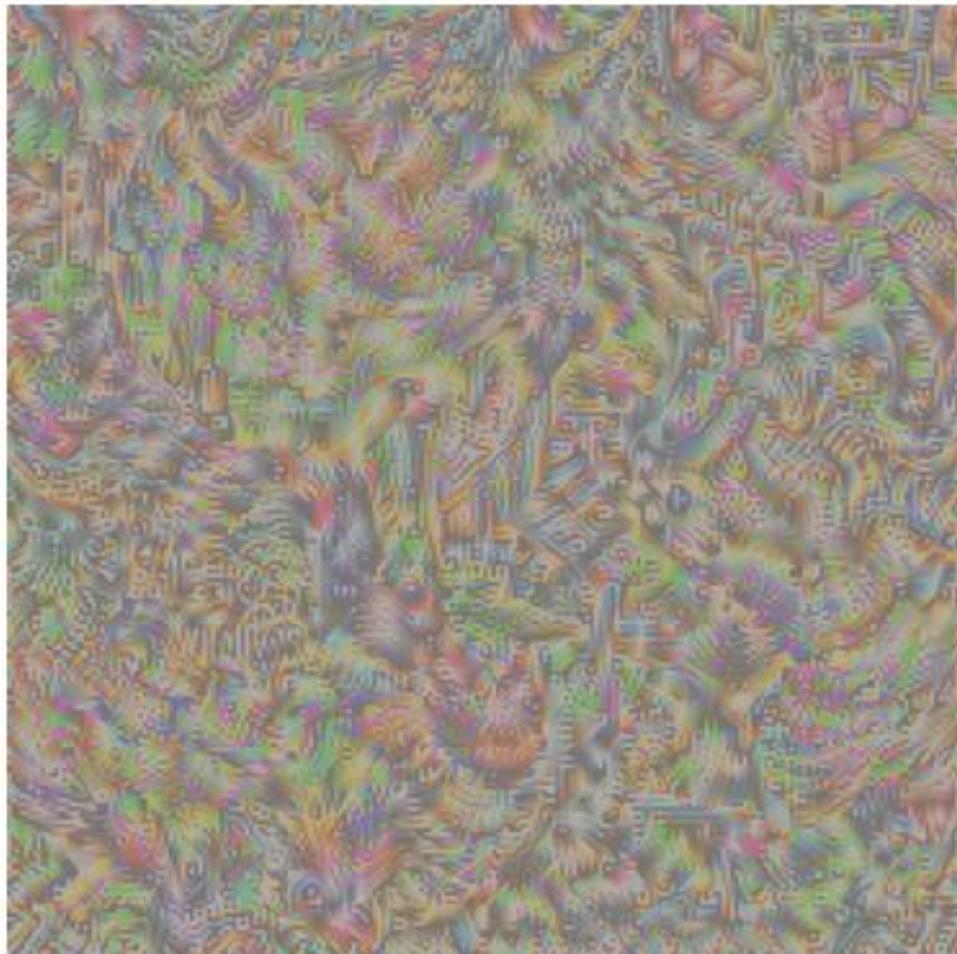
_ 1

در این بخش برای پیاده‌سازی روش Activation Maximization جهت تولید تصویری که مقدار logits کلاس "Hen" را بیشینه کند، از مدل VGG16 از پیش‌آموزش‌دیده استفاده کرده ام . در این روش، داده ورودی (تصویر) به صورت تدریجی بهینه‌سازی می‌شود تا مقدار logits کلاس هدف کلاس ("Hen") حداکثر شود.

توجه : کد های این مرحله را در یک فایل جوپیتر نوت بوک به نام FEATURE_VISUALIZATION نوشته ام

تصویر خروجی از کد به این گونه شد

Activation Maximization for Hen Class



- 2

همانطور که مشاهده میشود تصویر خروجی تصویر با معنایی نشده است که دلایل زیر را دارد :

الف - عدم استفاده از تکنیک های منظم سازی :

روش Activation maximization فقط بر روی بیشینه‌سازی فعال‌سازی یک نورون خاص یا خروجی یک کلاس خاص (که در اینجا کلاس مرغ است) تمرکز دارد بدون اینکه از روش‌های منظم‌سازی استفاده کند. این مسئله می‌تواند به تولید تصاویر بسیار نویزی و بدون ساختار منجر شود که به اشیای معنی‌دار شباهت ندارند.

ب : عدم وجود محدودیت‌های ویژگی‌های سطح بالا

در شبکه‌های عصبی کانولوشنی (CNN)، لایه‌های پایین ویژگی‌های پایه (مانند لبه‌ها و بافت‌ها) را ضبط می‌کنند، در حالی که لایه‌های بالاتر ویژگی‌های پیچیده‌تر (مانند اشکال و اشیاء) را ضبط می‌کنند. اغلب به طور مؤثری از این ویژگی‌های سطح بالا استفاده نمی‌کند، که منجر به تولید تصاویری می‌شود که فعال‌سازی نورون را بیشینه می‌کنند اما فاقد ساختار همگون هستند. استفاده از محدودیت‌های ویژگی‌های سطح بالا یا مدل‌های مولد می‌تواند به تولید تصاویر معنادارتر کمک کند.

ج : نویز گرادیان و بیشینه‌سازی بیش از حد

فرآیند بهینه‌سازی ممکن است نویز فرکانس بالا را تقویت کند به دلیل ماهیت بهروزرسانی‌های مبتنی بر گرادیان، که باعث می‌شود تصاویر تولید شده نویزی و انتزاعی به نظر برسند. این پدیده به این دلیل رخ می‌دهد که گرادیان‌های مورد استفاده برای بهروزرسانی تصویر می‌توانند ویژگی‌های بسیار ریز و اغلب نامربوط را در فضای ورودی برجسته کنند.

- 3

در این قسمت برای اینکه تصاویر با معنایی را برای کلاس Hen تولید کنم از تکنیک‌های مختلفی استفاده کرده ام که در ادامه آن‌ها را توضیخ میدهم

در کد از تکنیک Total Variation Regularization استفاده کرده ام که این تکنیک به طور موثر نویز‌های موجود در تصویر را کاهش میدهد در حالی که لبه‌های مهم و جزئیات تصویر را حفظ می‌کند. این ویژگی بسیار مهم است زیرا نویز می‌تواند به تصویر مصنوعی افزوده شود و باعث ایجاد الگوهای بی‌معنی شود. با کاهش نویز، تصویر صاف‌تر و طبیعی‌تر به نظر می‌رسد.

همچنین از تکنیک Random Shift هم استفاده کرده ام که باعث افزایش تنوع داده می‌شود زیرا جابجایی تصادفی پیکسل‌ها در هر گام بهینه‌سازی باعث می‌شود که مدل به تنوع بیشتری از ورودی‌ها مواجه شود. این کمک می‌کند که مدل از گیر افتادن در مینیمم‌های محلی جلوگیری کند و تصاویر با کیفیت‌تری تولید شود.

همچنین Random Shift به جلوگیری از الگوهای تکراری هم کمک میکند زیرا بدون جابجایی تصادفی، الگوریتم بهینهسازی ممکن است به الگوهای تکراری و مصنوعی گیر کند. جابجایی تصادفی به شکستن این الگوهای تکراری کمک میکند و باعث تولید تصاویری با جزئیات متنوع‌تر و واقعی‌تر می‌شود.

توضیحاتی درباره Total Variation Regularization برای داده‌های تصویری

یک تکنیک رایج در پردازش تصویر و یادگیری ماشین است Total Variation Regularization که هدف آن کاهش نویز در تصاویر و بهبود کیفیت بازسازی تصاویر است. این تکنیک به ویژه در مشکلات بازسازی تصویر مانند رفع نویز (denoising) و بازسازی تصاویر فشرده (compressed sensing) بسیار مؤثر است.

ایده اصلی Total Variation Regularization این است که با حداقل کردن تغییرات بین پیکسل‌های مجاور، تصویر را صاف‌تر کنیم و نویزهای جزئی را کاهش دهیم. این کار با استفاده از یک تابع هزینه انجام می‌شود که مجموع تغییرات بین پیکسل‌های مجاور را محاسبه می‌کند. این تابع هزینه به صورت زیر تعریف می‌شود:

$$TV(I) = \sum_{i,j} \sqrt{(\partial_x I_{i,j})^2 + (\partial_y I_{i,j})^2}$$

که در آن:

I تصویر مورد نظر است.

∂_x تغییرات تصویر در جهت افقی است

∂_y تغییرات تصویر در جهت عمودی است

کد های این قسمت را در همان فایل FEATURE_VISUALIZATION نوشته ام

تکنیک Total Variation Regularization را در کد به اینگونه پیاده سازی کردم.

```
# Total Variation Regularization
def total_variation_loss(img):
    tv_h = torch.pow(img[:, :, 1:, :] - img[:, :, :-1, :], 2).sum()
    tv_w = torch.pow(img[:, :, :, 1:] - img[:, :, :, :-1], 2).sum()
    return (tv_h + tv_w)
```

همچنین تکنیک Random shift را به اینگونه پیاده سازی کردم.

```
# Random shift
def random_shift(img, max_shift=5):
    _, _, h, w = img.size()
    shift_x, shift_y = np.random.randint(-max_shift, max_shift+1, size=2)
    img = torch.roll(img, shifts=(shift_x, shift_y), dims=(2, 3))
    return img
```

توجه : در کد علاوه بر دو روش بالا برای تولید تصاویر بهتر از روش DeepDream regularization

هم استفاده کرده ام

توضیح روش :

DeepDream تکنیکی است که توسط گوگل برای بصری سازی ویژگی ها و فعال سازی های داخلی شبکه های عصبی کانولوشنی (CNN) توسعه یافته است. این تکنیک الگوها را در یک تصویر تقویت و برجسته می کند تا تصاویر بهتر ایجاد کند. فرآیند DeepDream شامل اصلاح تکراری یک تصویر برای بیشینه سازی فعال سازی های برخی نورون ها یا لایه های شبکه است.

هدف اصلی DeepDream این است که فعال‌سازی‌های لایه‌های خاص یا نورون‌های خاصی در شبکه عصبی را بیشینه کند. با این کار، ویژگی‌هایی که شبکه یاد گرفته است، برجسته و قابل مشاهده می‌شوند.

هسته الگوریتم DeepDream شامل اجرای صعود گرادیان بر روی تصویر ورودی نسبت به فعال‌سازی‌های یک لایه انتخاب شده است. این بدان معناست که تصویر به گونه‌ای اصلاح می‌شود که فعال‌سازی نورون‌های هدف افزایش یابد.

اصلاح تصویر به صورت تکراری انجام می‌شود. در هر تکرار، تصویر کمی اصلاح می‌شود تا فعال‌سازی‌ها بیشتر تقویت شوند. این فرآیند تکراری منجر به ظاهر شدن تدریجی الگوهای ویژگی‌ها می‌شود.

کد DeepDream را به این شکل نوشته ام

```
# DeepDream regularization
def deepdream_loss(img):
    kernel = torch.tensor([[1, 4, 6, 4, 1],
                          [4, 16, 24, 16, 4],
                          [6, 24, 36, 24, 6],
                          [4, 16, 24, 16, 4],
                          [1, 4, 6, 4, 1]], dtype=torch.float32)
    kernel = kernel.unsqueeze(0).unsqueeze(0) / 256.0
    kernel = kernel.repeat(3, 1, 1, 1).to(img.device)
    img = torch.nn.functional.conv2d(img, kernel, padding=2, groups=3)
    return torch.sum(torch.pow(img, 2))
```

بنابراین من در کدی که نوشتم از سه تکنیک استفاده کرده ام:

Total Variation Regularization

Random shift

DeepDream regularization

با تکنیک‌های گفته شده و با تنظیم دقیق پارامترها سعی کردم ۵ تصویر مناسب را تولید کنم که به اینگونه شد.

Image 1

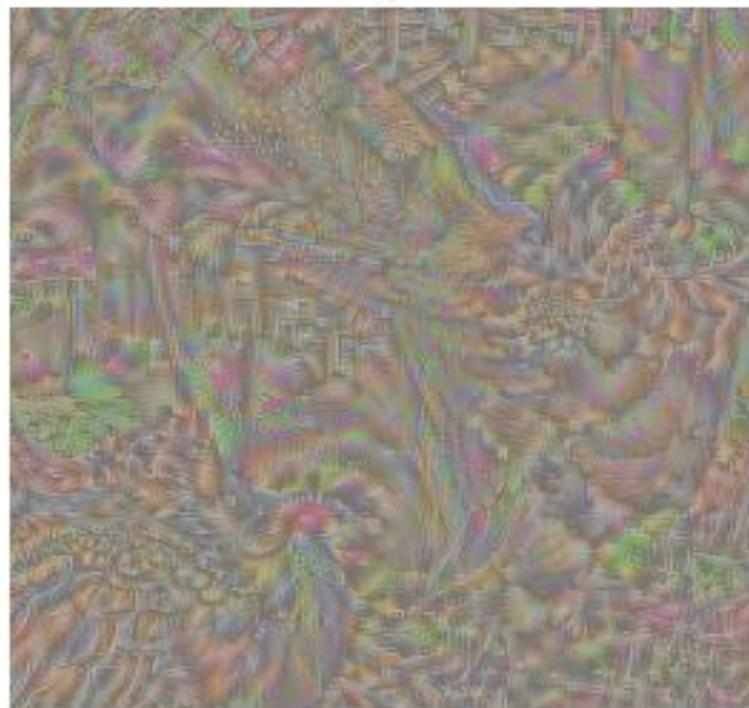


Image 2

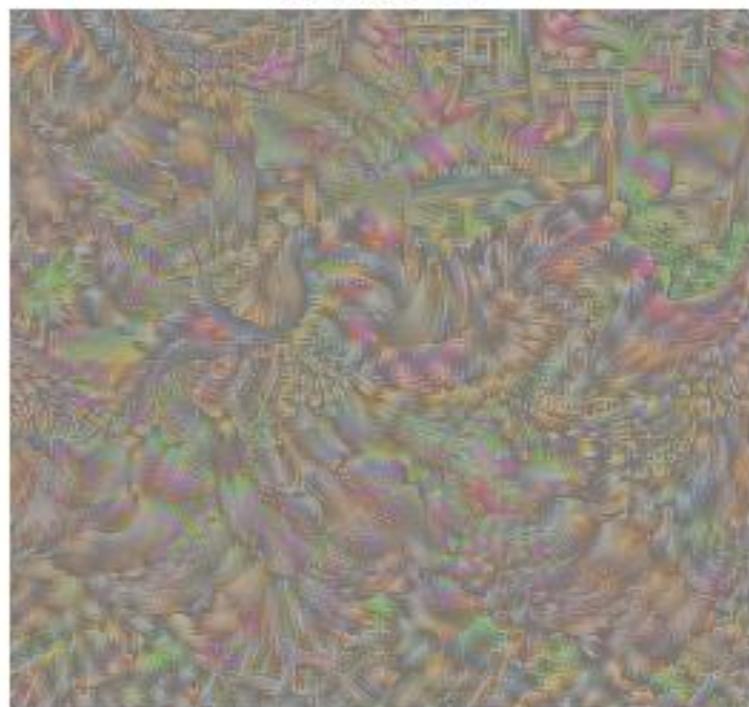


Image 3

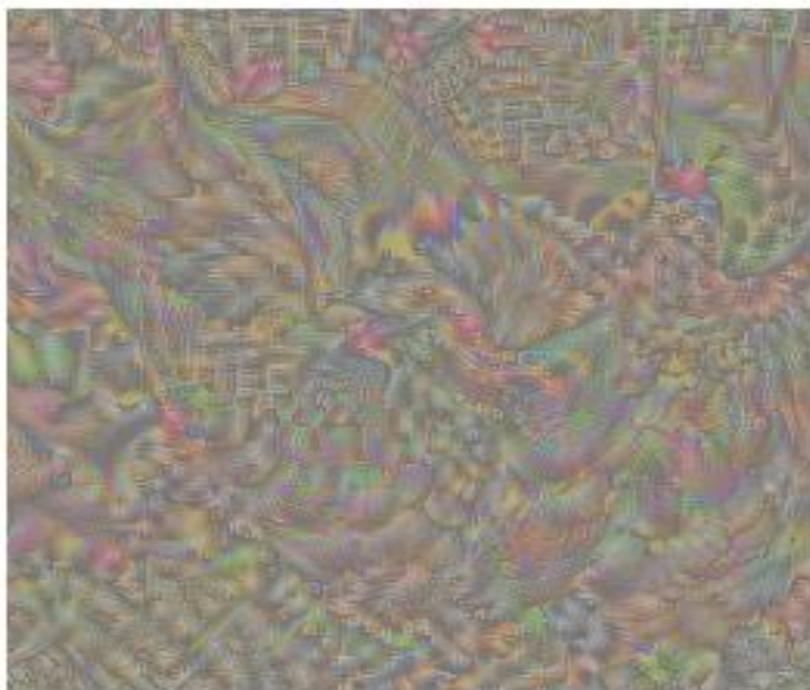


Image 4

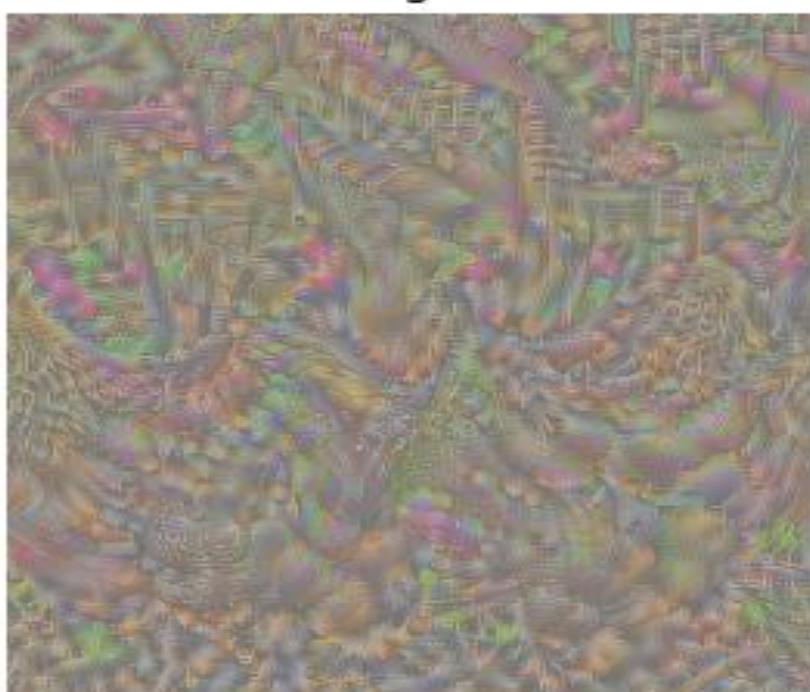
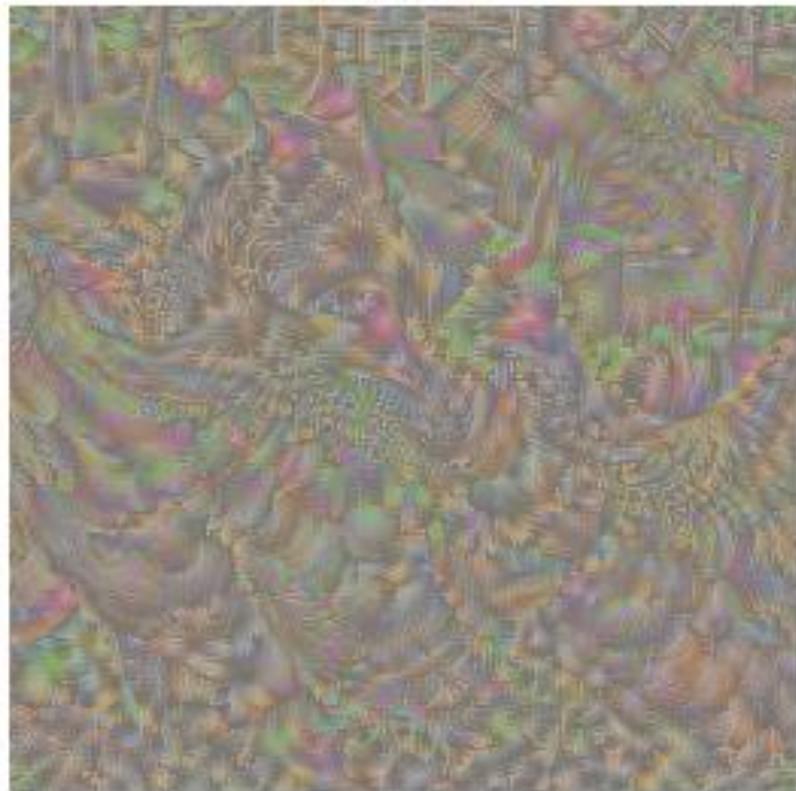


Image 5



تحلیل :

همانطور که مشاهده میشود در تصاویر نشانه هایی از مرغ دیده میشود مانند پر یا سر مرغ در تصاویر پیداست و نتیجه میشود روش هایی که اعمال شده نتیجه داده است . من سعی و خطاهای زیادی با روش ها و هایپر پارامتر های زیادی انجام دادم و در نهایت بهترین نتیجه خودم را قرار دادم .