به نام خدا

دانشکده مهندسی برق

گزارش درس یادگیری ماشین

مقطع: کارشناسی ارشد گرایش: مهندسی کنترل

گزارش آزمون میانترم

توسط:

مرجان محمدی

۴۰۱۱۱۵۳۴

استاد درس:

دکتر علیاری

لینک کولب

لینک گیت هاب

بهار ۱۴۰۳

۱ پرسش یک

درستی یا نادرستی هریک از گزاره‌های زیر را با ذکر دلیل مشخص کنید.

۱. طبقه‌بند بیز، بهترین طبقه‌بندی است که می‌توان برای جداسازی یک مسألهٔ دوکلاسه طراحی کرد.

۲. استفاده از رویکرد بیز برای تخمین پارامترهای توزیع، می‌تواند مانع از بیش‌برازش (Overfitting) شود.

۳. استفاده از معیار Information Gain برای ساخت درخت در شرایطی که بعضی از ویژگی‌ها حالات زیادی دارند، مناسب نیست.

۴. هر شبکهٔ عصبی چندلایه با توابع فعال‌ساز خطی در لایه‌های پنهان می‌تواند به عنوان یک شبکهٔ عصبی بدون هیچ لایه پنهانی نمایش داده شود.

بخش اول)

به صورت کلی این گزاره نمیتواند درست باشد چونکه طبقه بند بیز فرض میکند که ویژگی هایی که داخل یک کلاس هستن نسبت به هم کامل مستقل اند در صورتی که در داده های دنیای واقعی نمیتونه درصت باشه و این الگوریتم باعث می شود که دقت طبقه بند خوب نباشه.

همچنین وقتی پای مدل‌های مختلف به میان می‌آید، نمی‌شه یکی رو برتر دونست بدون اینکه به خصوصیات داده‌ها توجه کنیم. مثلاً، شبکه‌های عصبی، SVM، و درخت تصمیم در شرایط خاصی می‌توانند کارآیی بهتری داشته باشند.

طبقه‌بند بیز وقتی خیلی خوب جواب می‌ده که داده‌ها به خوبی فرضیات اون رو تأیید کنن. ولی اگه داده‌ها پیچیده‌تر باشن یا بین ویژگی‌ها وابستگی‌های قوی وجود داشته باشه، این مدل ممکنه کارایی لازم رو نداشته باشه و همچنین ، برخی مدل‌های پیچیده‌تر مثل شبکه‌های عصبی می‌تونن الگوهای پیچیده‌تری رو یاد بگیرن، به خصوص در داده‌های با ویژگی‌های غیرخطی. پس، گرچه طبقه‌بند بیز ممکنه در بعضی مواقع خیلی خوب کار کنه، نمی‌توان اون رو به عنوان بهترین گزینه برای همه موارد دوکلاسه دانست. همه چیز به خصوصیات مسئله، توزیع داده‌ها و نیازهای خاص پروژه بستگی داره.

بخش دوم)

درسته چون استفاده از روش بیز برای تخمین پارامترهای توزیع خیلی کمک می‌کنه که مدل‌ها دچار بیش‌برازش نشن. دلیلش هم اینه که:

- توی روش بیز، ما از اول یه سری حدس‌ها درباره‌ی پارامترها داریم که اینا بر اساس تجربیات قبلی‌مونه. این حدس‌ها مثل یه تنظیم‌کننده عمل می‌کنن و اجازه نمی‌دن مدل خیلی بچسبه به داده‌های آموزشی که ممکنه شامل خطا یا نویز باشن.

- توی این روش، پارامترها همواره بر اساس داده‌های جدید دوباره تنظیم می‌شن. این یعنی پارامترها فقط به اندازه کافی تغییر می‌کنن که با داده‌ها جور دربیان و از تغییرات شدیدی که ممکنه بیش‌برازش ایجاد کنن، جلوگیری می‌شه.

- این توزیع که از ترکیب احتمالات قبلی و شواهد جدید حاصل می‌شه، بهمون یه تصویر کامل از وضعیت پارامترها می‌ده که هم داده‌های دیده شده رو در بر می‌گیره و هم اطلاعات قبلی‌مون رو. این کمک می‌کنه که تخمین‌های دقیق‌تری داشته باشیم و از تصمیم‌گیری‌های افراطی بر پایه داده‌های کم یا پر از نویز پرهیز کنیم.

خلاصه که روش بیزی با اینکه از تجربه‌های قبلیمون استفاده می‌کنه و پارامترها رو مرتب با داده‌های جدید تنظیم می‌کنه، می‌تونه خیلی کمک کنه که مدل‌هامون دقیق‌تر و بدون اشتباه از بیش‌برازش بمونن.

بخش سوم)

استفاده از Information Gain واسه ساخت درخت تصمیم، وقتی که یه سری ویژگی‌ها حالت‌های زیادی دارن، چندان کار درستی نیست چون:

Information Gain یه شاخصه که نشون می‌ده یه ویژگی چقدر می‌تونه کمک کنه تا داده‌ها رو تفکیک کنیم و به کلاس‌های مختلف بفرستیم. این شاخص بر اساس یه چیزی به اسم Entropy سنجیده می‌شه که هرچی پایین‌تر باشه، یعنی داده‌ها کمتر به هم ریخته و مرتب‌تر هستن.

اما وقتی یه ویژگی خیلی زیاد حالت داره، این شاخص ممکنه درست به ما اصلاعات نده. مثلاً وقتی یه ویژگی داریم که صد حالت مختلف داره. این ویژگی می‌تونه داده‌ها رو به صد دسته کوچیک تقسیم کنه که هر دسته شاید فقط یکی دو نمونه داشته باشه. اینجوری که پیش می‌ریم، در نهایت مدلمون فقط روی داده‌های آموزشی خوب کار می‌کنه و وقتی بخوایم ازش توی دنیای واقعی استفاده کنیم، درست کار نمیکنه چون نمی‌تونه خوب تعمیم پیدا کنه.

پس، در نتیجه، اگه می‌خوایم از Information Gain استفاده کنیم، باید مواظب باشیم که ویژگی‌هایی که حالت زیادی دارند ممکنه باعث بیش‌برازش بشن و به مدل ضربه بزنن.

بخش چهارم)

بله. وقتی یه شبکه عصبی داریم که توی لایه‌های پنهونش فقط از توابع فعالسازی خطی استفاده می‌کنه، می‌تونیم بگیم این شبکه همون کاری رو می‌کنه که یه شبکه بدون هیچ لایه پنهونی می‌تونه بکنه. چرا؟ چون

وقتی توابع فعالسازمون خطی هستن، هرچیزی که توی لایه‌ها اتفاق می‌افته، فقط یه سری جمع و ضرب ساده‌اس. این یعنی تموم این لایه‌های وسطی که داریم رو می‌شه فشرده کرد و در اصل تبدیلشون کرد به یک لایه خطی ساده.

خلاصه‌اش اینه که وقتی همه توابع فعالسازی خطی‌ان، نیازی نیست چند لایه داشته باشیم چون همه‌شونو می‌شه در یک لایه خلاصه کرد. این مدل به نظر می‌رسه خیلی پیچیده‌اس، در واقع خیلی ساده‌تره.

۲ پرسش دو

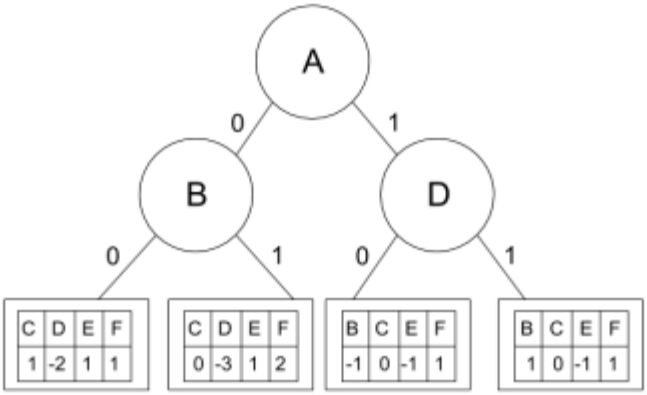
برای بهره‌برداری از ویژگی‌های مطلوب طبقه‌بندهای درخت تصمیم و پرسپترون، سارا الگوریتم جدیدی به نام «درخت پرسپترون» ایجاد کرده که ویژگی‌های هر دو را ترکیب می‌کند. درخت‌های پرسپترونی شبیه به درخت‌های تصمیم هستند؛ اما هر leaf node به‌جای مکانیزم رأی اکثریت، شامل یک پرسپترون است.

برای ایجاد یک درخت پرسپترون؛ اولین مرحله، اجرای یک الگوریتم یادگیری درخت تصمیم معمولی (مانند ID3) و انجام تقسیم‌بندی بر اساس ویژگی‌ها تا رسیدن به عمق حداکثر مشخص شده است. هنگامی که به عمق حداکثر می‌رسیم، در هر leaf node، یک پرسپترون روی ویژگی‌های باقی‌مانده که هنوز در آن شاخه استفاده نشده‌اند، آموزش داده می‌شود. دسته‌بندی یک نمونه جدید از طریق مراحل مشابهی انجام می‌شود. ابتدا نمونه از طریق درخت تصمیم بر اساس مقادیر ویژگی‌هایش گذر می‌کند. وقتی به یک leaf node می‌رسد، پیش‌بینی نهایی با اجرای پرسپترون متناظر در آن گره انجام می‌شود.

فرض کنید که دارای مجموعه داده‌ای با ۶ ویژگی دودویی $\{A, B, C, D, E, F\}$ و دو برچسب خروجی $\{-1, 1\}$ هستید. یک درخت پرسپترون با عمق ۲ روی این مجموعه داده‌ها در شکل ۱ آمده است. وزن‌های پرسپترون نیز در leafها آمده است (فرض کنید که بایاس برای هر پرسپترون $b = 1$ است).

۱. برای نمونهٔ $\mathbf{x} = [1, 1, 0, 1, 0, 1]$، درخت پرسپترون داده‌شده چه برچسب خروجی‌ای را پیش‌بینی می‌کند؟

۲. آیا مرزتصمیم درخت پرسپترون همواره خطی است؟ برای مقادیر کوچک حداکثر عمق، کیفیت آموزش درخت تصمیم و درخت پرپسترون را با ذکر دلیل مقایسه کنید. آیا تفاوتی دارند؟



شکل ۱: درخت پرسپترون با عمق دو.

بخش اول)

فرض کنیم یه داده داریم به اسم x = [1, 1, 0, 1, 0, 1] و می‌خواهیم ببینیم برچسب خروجیش توی درخت پرسپترون با عمق ۲ چی می‌شه. اول این داده رو می‌فرستیم از ریشه‌ی درخت. چون ویژگی A مقدارش ۱ هست، پس می‌ریم سمت راست. بعد ویژگی B هم که ۱ هست، پس می‌رسیم به یه گره که توش یه پرسپترون هست با وزن‌های w = [2, -1, 1, -2]حالا برای محاسبه خروجی پرسپترون باید حساب کنیم w.x + b که می‌شه:

$$w.x + b = (2 * 1) + (-1 * 0) + (1 * 1) + (-2 * 0) + 1 = 3$$

چون این عدد مثبته، پرسپترون میگه برچسب +۱. پس برچسبی که برای این داده پیش‌بینی می‌کنه مثبت یک می‌شه.

بخش دوم)

حالا در مورد مرز تصمیم در درخت پرسپترون، این مرز لزوماً خطی نیست. تو هر گره انتهایی، یه پرسپترون هست که یه مرز تصمیم خطی می‌سازه. ولی وقتی این مرزها رو با هم ترکیب می‌کنیم توی کل درخت، یه مرز تصمیم غیرخطی پیچیده‌تر درست می‌شه. پس مرز نهایی درخت پرسپترون، یه ترکیبی از خطی و غیرخطیه که به ساختار درخت و وزن‌های پرسپترون‌ها بستگی داره.

وقتی صحبت از مقایسه کیفیت آموزش درخت تصمیم و درخت پرسپترون برای عمق‌های کوچک می‌شه:

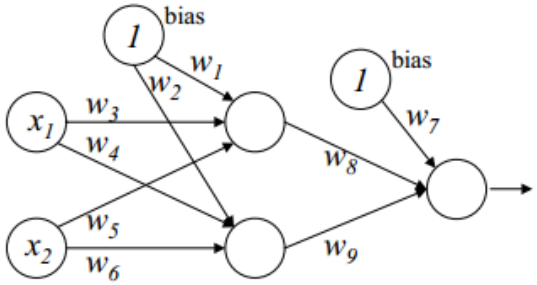
درخت تصمیم تو عمق‌های کوچک، فقط می‌تونه تقسیم‌بندی‌های ساده‌ای رو روی تعدادی محدود از ویژگی‌ها انجام بده، پس ممکنه نتونه الگوهای پیچیده‌تری رو یاد بگیره و کارایی‌ش محدود بمونه.

ولی توی درخت پرسپترون این درختا با پرسپترون‌های تو گره‌های انتهایی، می‌تونن ترکیبات خطی از ویژگی‌های باقی‌مانده رو یاد بگیرن، که این بهشون اجازه می‌ده مرزهای تصمیم پیچیده‌تری رو نسبت به درخت تصمیم مدل‌سازی کنن. پس احتمالاً در عمق‌های کوچک، عملکرد بهتری نسبت به درخت تصمیم دارن، چون می‌تونن از ویژگی‌های بیشتری استفاده کنن.

ولی خب، باید دقت کرد که تو عمق‌های بزرگ‌تر، درخت تصمیم می‌تونه تقسیم‌بندی‌های پیچیده‌تری رو انجام بده و عملکردش به درخت پرسپترون نزدیک‌تر بشه. در نهایت، انتخاب بین این دو مدل به ویژگی‌های داده، پیچیدگی مسئله و محدودیت‌های محاسباتی بستگی داره.

۳ پرسش سه

شبکهٔ عصبی آوردهشده در شکل ۲ را برای یک مسألهٔ طبقهبندی دوکلاسه در نظر بگیرید. فرض کنید که لایههای میانی از تابع فعالساز خطی ($h(z) = cz$) و لایهٔ خروجی از تابع سیگموئید ($g(z) = \frac{1}{1+e^{-z}}$) استفاده میکند. این شبکه میخواهد یک تابع برای $P(Y = 1 \mid X, w)$ که در آن $X = (x_1, x_2)$ و $W = (w_1, w_2, \ldots, w_9)$ است را یاد بگیرد.



شکل ۲: شبکهٔ عصبی سوال سوم.

۱. خروجی شبکهٔ عصبی $P(Y = 1 \mid X, w)$ را بر حسب پارامترهای شبکه (W, x) و ثابت c نوشته و مرز تصمیم نهایی را به دست آورید.

۲. آیا میتوان یک شبکهٔ عصبی بدون لایهٔ مخفی به دست آورد که معادل شبکهٔ عصبی فوق باشد؟ در صورت وجود، شبکهٔ پیشنهادیتان را رسم کنید.

بخش اول)

وقتی می‌خوایم بفهمیم خروجی شبکه عصبی چیه و کجا خط تصمیم می‌افته، باید مسیر رو از ورودی تا خروجی دنبال کنیم.

حالا فرض کنیم خروجی‌های لایه میانی رو داریم، می‌گن به اونا h1 و h2چون توی لایه میانی با تابع فعال‌سازی خطی کار می‌کنیم، داریم:

h1 = c(w3 * x1 + w5 * x2 + w1 * b)

h2 = c(w4 * x1 + w6 * x2 + w2 * b)

حالا برای خروجی نهایی، از تابع سیگموئید استفاده می‌کنیم. پس می‌شه:

P(Y=1|X,W) = g(w8 * h1 + w9 * h2 + w7 * b) = 1/ 1 + exp(-(w8(c(w3 * x1 + w5 * x2 + w1 * b)) * + w9 * (c(w4 * x1 + w6 * x2 + w2 * b)) + w7 * b) = 1/ 1 + exp(-(w8 * h1 + w9 * h2 + w7 * b)

حالا برای مرز تصمیم که برای طبقه‌بندی دودوییه، جایی که P(Y=1|X,W) بشه ۰,۵، معادله‌اش اینجوری می‌شه:

W8 * c(w3 * x1 + w5 * x2 + w1 * 1) + w7c(w4 * x1 + w6 * x2 + w2 * 1) + w7 * 1 = 0

این معادله خط تصمیم نهایی رو تو فضای ویژگی‌ها (x1, x2) نشون می‌ده که یه خط غیرخطیه چون اثرات تعاملی بین x1 و x2 رو داره.

بخش دوم)

اگه بخوایم یه شبکه عصبی بدون لایه مخفی واسه شبکه عصبی که داریم طراحی کنیم، این کار رو می‌تونیم بکنیم. دلیلش هم اینه که از نظر جبری، می‌شه هر شبکه‌ای که لایه‌های مخفی داره رو با یک شبکه تک لایه که تعداد گره‌های کافی داشته باشه جور کرد.

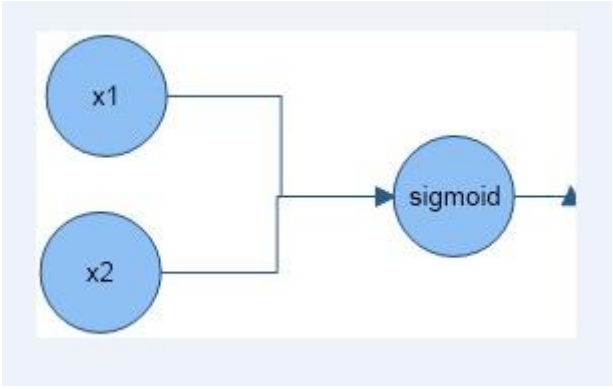
فرض کن این شبکه پیشنهادی بدون لایه مخفی به این شکله که ورودی‌های x1 و x2 مستقیم می‌رن به یه گره تو لایه خروجی. تابع فعال‌سازی که اینجا استفاده می‌کنیم همون تابع سیگموئیده. خروجی شبکه‌مون که با P(Y=1|X,W) نشون می‌دیم، همون خروجی شبکه اصلیه که لایه مخفی داشت.

حالا خروجی این شبکه تک لایه به این شکل می‌شه:

P(Y=1|X,W) = g(w0 + w1 * x1 + w2 * x2 + w3* x1x2 + w4 * x1^2 + w5 * x2^2)

اینجا W که می‌شه (w0, w1, w2, w3, w4, w5) وزن‌های شبکه هستند و باید از داده‌های آموزشی یاد گرفته بشن.

یه مزیت این شبکه تک لایه اینه که خیلی ساده‌تر از شبکه اصلیه که لایه مخفی داشت. ولی با این حال هنوز می‌تونه مرزهای تصمیم غیرخطی پیچیده‌ای رو مدل کنه. البته باید یادمون باشه که پیدا کردن بهترین مجموعه وزن‌ها تو این شبکه ممکنه کمی سخت‌تر از شبکه اصلی باشه.



سوال چهارم)

ابتدا طبق زیر دیتا را دانلود می کنیم و میخوانیم

```
!pip install --upgrade --no-cach-dir gdown
! gdown 1eX7Mr1C1LTraVV5ju7hQEkAFBH0-xLJp
```

```
dataset = sio.loadmat('/content/DATA.mat')
data_NOV9 = pd.DataFrame(dataset['NOV9'])
data_NOV17 = pd.DataFrame(dataset['NOV17'])
```

سپس طبق جدول ۲ دیتا ها را به دو قسمت نرمال و فالتی تقسیم کرده و لیبل میزنیم. نرمال صفر و فالتی ۱

سپس ۹ نوامبر را به عنوان دیتای ترین و ۱۷ نوامبر را به عنوان دیتای تست و ولیدیشن در نظر میگیریم.

```
# Initialize labels as 0 for all samples
data_NOV9['label'] = 0
data_NOV17['label'] = 0

# Assign label 1 to specific segments
data_NOV9.loc[57275:57550, 'label'] = 1
data_NOV9.loc[58830:58930, 'label'] = 1
data_NOV9.loc[58520:58625, 'label'] = 1

data_NOV17.loc[54600:54700, 'label'] = 1
data_NOV17.loc[56670:56770, 'label'] = 1

# Display a few samples from the data to confirm the labeling
print(data_NOV9.head())
print(data_NOV17.head())
```

دیتای ما مقدار nan ندارد و فقط آن را نرمال می کنیم.

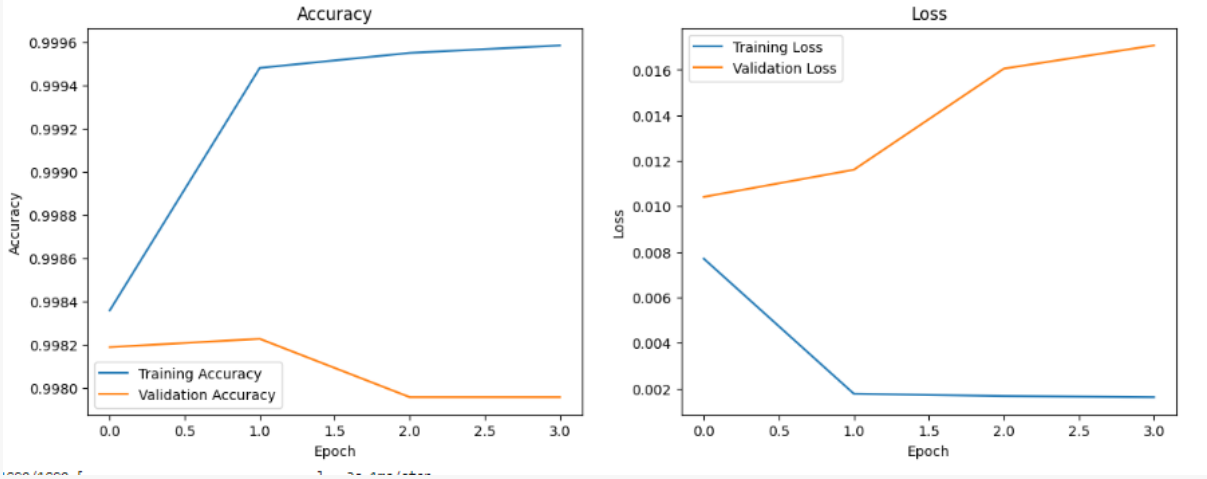
از شبکه عصبی ۳ لایه با تابع فعالساز relu استفاده میکنیم.از early stop استفاده کردیم.هایچر چارامترها و مدل در زیر است:تعداد ایپاک ۵۰

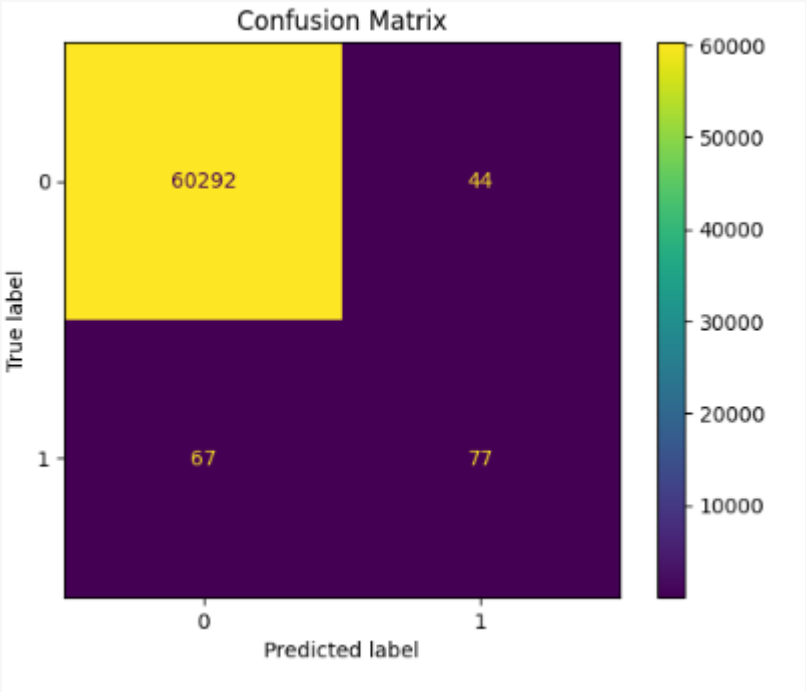
```
# Define model
model = Sequential([
    Dense(128, activation='relu',
input_shape=(train_features.shape[1],)),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])
```

```
# Early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3)


# Train model
history = model.fit(train_features, train_labels, epochs=50,
batch_size=32,
                    validation_data=(validation_features,
validation_labels), callbacks=[early_stopping])
```





چون کلاس های ما بالانس نیست این اتفاق در ماتریس در هم ریختگی افتاده است.و برای جلوگیری از اورفیت در ایپاک ۴ استپ شده است با استفاده از ارلی استاپ دقت ولیدیشن به ۹۹ درصد رسیده است.
☹به دلیل وقت کم و قطعی فیلتر شکن تا همینجا تونستم انجام بدم

```
100% 2.91M/2.91M [00:00<00:00, 149MB/s]
Epoch 1/50
2700/2700 [==============================] - 21s 7ms/step - loss: 0.0077 - accuracy: 0.9984 - val_loss: 0.0104 - val_accuracy: 0.9982
Epoch 2/50
2700/2700 [==============================] - 8s 3ms/step - loss: 0.0018 - accuracy: 0.9995 - val_loss: 0.0116 - val_accuracy: 0.9982
Epoch 3/50
2700/2700 [==============================] - 7s 3ms/step - loss: 0.0017 - accuracy: 0.9995 - val_loss: 0.0161 - val_accuracy: 0.9980
Epoch 4/50
2700/2700 [==============================] - 9s 3ms/step - loss: 0.0016 - accuracy: 0.9996 - val_loss: 0.0171 - val_accuracy: 0.9980
1890/1890 [==============================] - 3s 1ms/step - loss: 0.0178 - accuracy: 0.9982
Test accuracy: 0.9981646537780762
```