به نام خدا

دانشگاه تهران

دانشکده مهندسی مکانیک

درس هوش مصنوعی

تمرین پنجم

نام و نام خانوادگی	محمد اخلاقی
شماره دانشجویی	۸۱۰۶۰۱۰۲۷
تاریخ ارسال گزارش	۱۴۰۲.۰۴,۰۱

فهرست

شکل‌ها

پاسخ ۱. مفاهیم پایه

الف)

برای classification:

۱- Categorical Cross-Entropy Loss

در هر دسته، این تابع محاسبه می‌کند که مدل چقدر به درستی توزیع پیش‌بینی شده را تشخیص داده است و چقدر از توزیع واقعی برچسب‌ها متفاوت است. با محاسبه اختلاف بین این دو توزیع و استفاده از لگاریتم نتیجه، این تابع زیان را بهبود می‌دهد.

$$CE\ Loss = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot log(p_{ij})$$

۲- Binary Cross-Entropy Loss

این تابع زیان برای اندازه‌گیری فاصله بین توزیع پیش‌بینی شده و توزیع واقعی برچسب‌ها استفاده می‌شود. با استفاده از لگاریتم نرمالیزه شده از احتمال پیش‌بینی شده برای دسته مثبت (با مقدار ۱) و احتمال پیش‌بینی شده برای دسته منفی (با مقدار ۰)، تابع زیان بین پیش‌بینی و واقعیت را اندازه‌گیری می‌کند.

$$CE\ Loss = \frac{1}{n}\sum_{i=1}^{N} -\ (y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)$$

برای regression

۱- Mean Squared Error Loss: ین تابع زیان میزان فاصله میان مقادیر پیش‌بینی شده و مقادیر واقعی هدف را می‌سنجد. با مربع‌کردن این فاصله، تابع زیان متوسط مربعات را محاسبه می‌کند.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

۲- Mean Absolute Error Loss:

این تابع زیان میزان فاصله میان مقادیر پیش‌بینی شده و مقادیر واقعی هدف را می‌سنجد. با مقدار مطلق کردن این فاصله، تابع زیان متوسط مطلق را محاسبه می‌کند.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}|$$

ب)

لایه Batch Normalization یک یه کلیدی در شبکه‌های عصبی عمیق است که بهبود عملکرد و سرعت آموزش شبکه را تسهیل می‌کند. این لایه به طور معمول بین لایه‌های پنهان شبکه قرار می‌گیرد.

عملکرد لایه نرمالیزاسیون دسته‌ای به این صورت است که در هر مرحله آموزش، ورودی‌ها را به شکل دسته‌های کوچکتر تقسیم می‌کند (معمولاً به صورت مینی‌بچ). سپس ویژگی‌های آماری مانند میانگین و انحراف معیار برای هر دسته استخراج می‌شوند. سپس این ویژگی‌های آماری را با استفاده از پارامترهای آموزشی تبدیل کرده و داده‌ها را نرمالیزه می‌کند. در نهایت، ورودی‌های نرمالیزه شده به لایه بعدی ارسال می‌شوند.

به طور کلی لایه نرمالیزاسیون دسته‌ای در شبکه‌های عصبی عمیق می‌تواند تاثیر مثبتی بر عملکرد و سرعت آموزش شبکه داشته باشد و موجب بهبود و استحکام عملکرد شبکه در حضور شبکه‌های عصبی عمیق شود.

پ)

لایه نرمالیزاسیون :(Normalization Layer) این لابه، به عنوان یک لایه مستقل در شبکه عمل می‌کند و وظیفه نرمالیزه کردن ورودی‌ها را بر عهده دارد. با استفاده از ویژگی‌های آماری مانند میانگین و انحراف معیار، ورودی‌ها را نرمالیزه می‌کند. این الیه عموماً قبل از لایه‌های فعال‌سازی مانند ReLU قرار می‌گیرد.

لایه گروهی نرمالیزاسیون :(Normalization Group) در این روش، ورودی‌ها را به گروه‌های کوچکتر تقسیم می‌کنند و ویژگی‌های آماری مانند میانگین و انحراف معیار را برای هر گروه محاسبه می‌کنند. با اعمال تبدیل‌های نرمالیزه بر روی هر گروه، ورودی‌ها را نرمالیزه می‌کنند.

تفاوت اصلی بین الیه نرمالیزاسیون دسته‌بندی و دو روش دیگر این است که الیه نرمالیزاسیون دسته‌بندی با استفاده از آمار **دسته** (میانگین و انحراف معیار دسته) عمل می‌کند، در حالی که لایه نرمالیزاسیون و گروهی نرمالیزاسیون ویژگی‌های آماری را برای هر **نمونه یا گروه محاسبه** می‌کنند.

پاسخ ۲ – دسته‌بندی تابلوهای راهنمایی رانندگی

الف) فراخوانی و پیش پردازش داده:

دادگان در گوگل درایو قرار گرفت و پس از متصل کردن آن به کولب پیشپردازش آغاز شد.

تابع پیشپردازش هر عکس را خوانده، سایزآن را ۳۰*۳۰ تبدیل کرده و با تقسیم کردن هر پیکسل بر ۲۵۵ آنها را استاندارد میکند.

```python
# Resize the images to 30x30x3 and divide by 255
def preprocess_image(image_path):
    image = cv2.imread(image_path)
    if image is not None:
        image = cv2.resize(image, (30, 30))
        image = image / 255.0
        return image
    else:
        print(f"Error loading image: {image_path}")
        return None
```

شکل ۱: تابع دریافت، استاندارد سازی و تغییر سایز تصویر

فایل csv مربوط به train را خوانده و مسیر عکسها را از ستون path برداشته و به کمک تابع preprocess_image هر عکس آماده شد. همچنین کلاس را از ستون classid استخراج کردیم و در دو لیست عکس ها و کلاس مربوط را به صورت متناظر ثبت کردیم.

```python
train_images = []
train_labels = []
for index, row in train_data.iterrows():
    print(index)
    image_path = os.path.join(dataset_path, row['Path'])
    image = preprocess_image(image_path)
    if image is not None:
        train_images.append(image)
        train_labels.append(row['ClassId'])
```

شکل ۲: دریافت تصاویر آموزش

داده های train به دو گروه train و val تقسیم شده و در نهایت لیست عکس‌ها به آرایه های نامپای تبدیل میشود.

```
# Divide the train data into train and validation sets
train_images, val_images, train_labels, val_labels = train_test_split(train_images,
                                                                      train_labels, test_size=0.1, random_state=42)
train_images = np.array(train_images)
val_images = np.array(val_images)
train_labels = np.array(train_labels)
val_labels = np.array(val_labels)
```

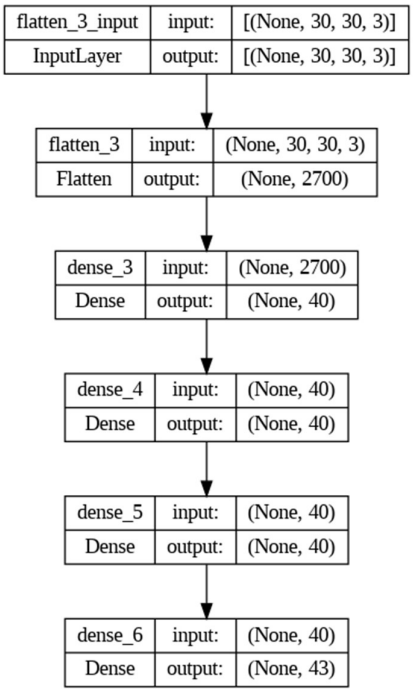
شکل ۳: تقسیم دادگان آموزش به آموزش و ولیدیشن

همین مراحل(به غیر از تقسیم داده ها به دو گروه) برای دادگان پوشه تست نیز انجام شد.

در نهایت دادگان به صورت numpy سیو شد تا سرعت لود کردن آنها سریع باشد.

معماری شبکه MLP:

شبکه MLP با ۳ لایه مخفی با ۴۰ نورون در هر لایه تعریف شد.



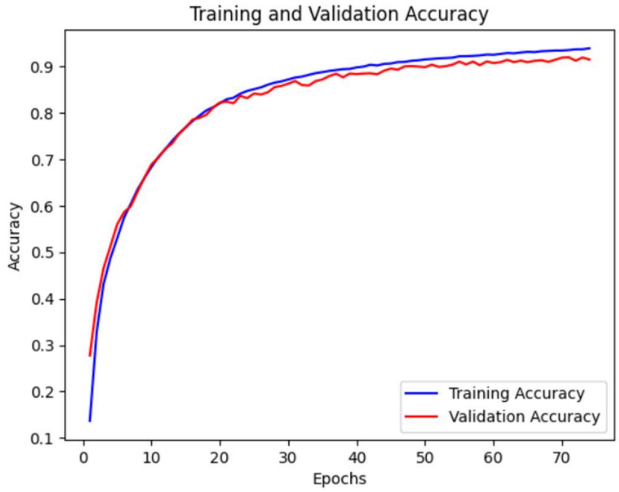
شکل ۴: رسم شبکه **MLP**

برای کامپایل کردن از بهینه ساز adam استفاده شد. و برای جلوگیری از overfit شدن شبکه early_stop بر اساس تغییرات val_loss تعریف شد. همچنین نرخ یادگیری را با آزمایش و سعی خطا روی ۰,۰۰۰۰۵ تنظیم کردیم. **نرخ یادگیری زیاد(به طور مثال ۰,۰۰۱) دقت نامناسبی را هم برای دادگان train و هم برای دادگان test به همراه داشت و نرخ یادگیری کوچکتر نیز باعث طولانی شدن زمان آموزش و بیشتر شدن ایپاک لازم برای رسیدن به دقت بالا میشود.** دقت نهایی بر روی داده های تست برابر ٪۸۲,۵ شد.

جدول ۱: دقت شبکه **MLP**

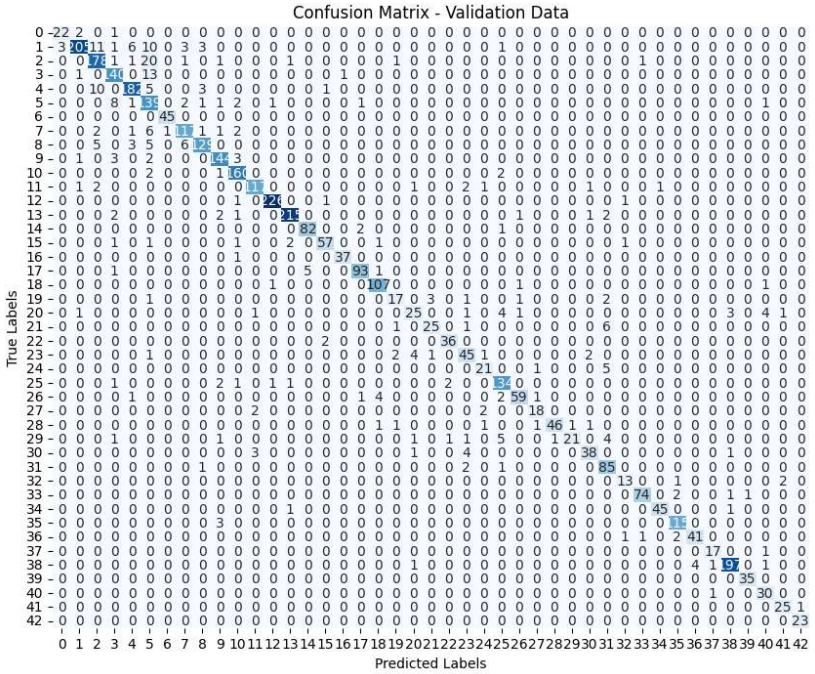
Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.8228	0.8568	0.9130	**0.3873**

دقت مدل بر روی داده های validation و train در هر ایپاک در نمودار زیر رسم شده است.

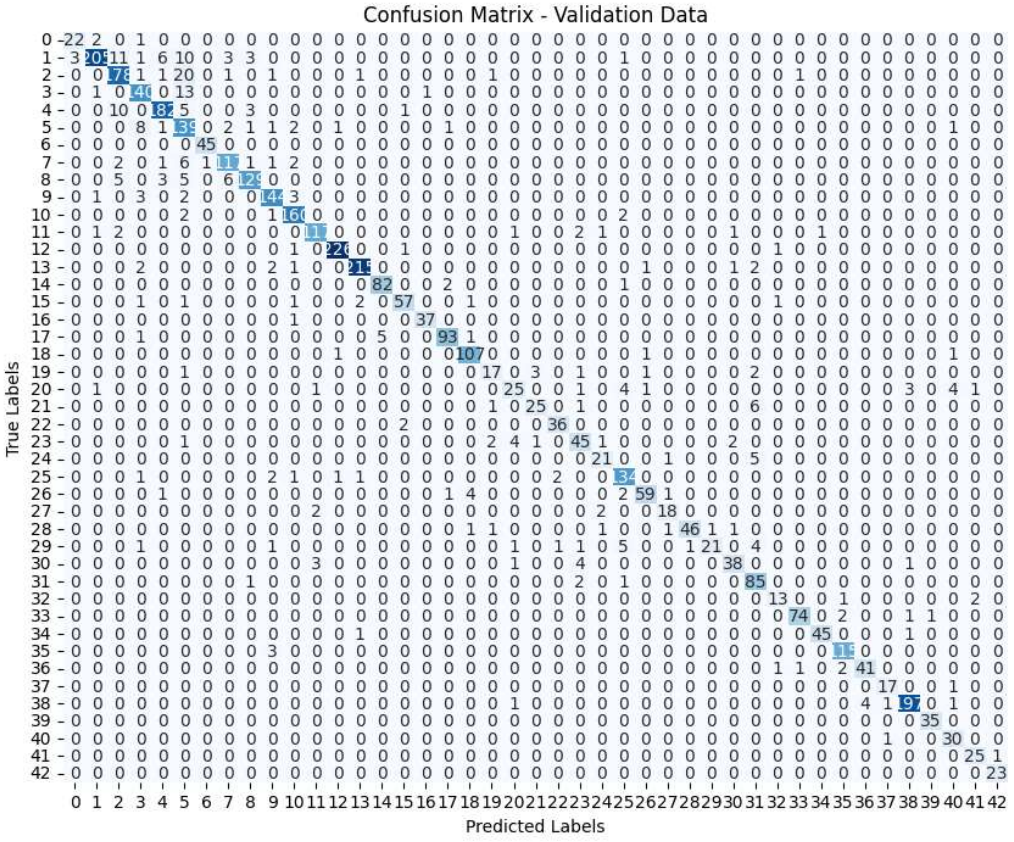


شکل ۵: دقت شبکه **mlp**

ماتریس سردرگمی هم رسم شد:



شکل ۶: ماتریس سردرگمی داده های **Val** شبکه‌ی **MLP**



شکل ۷:ماتریس سردرگمی داده های تست شبکه‌ی **MLP**

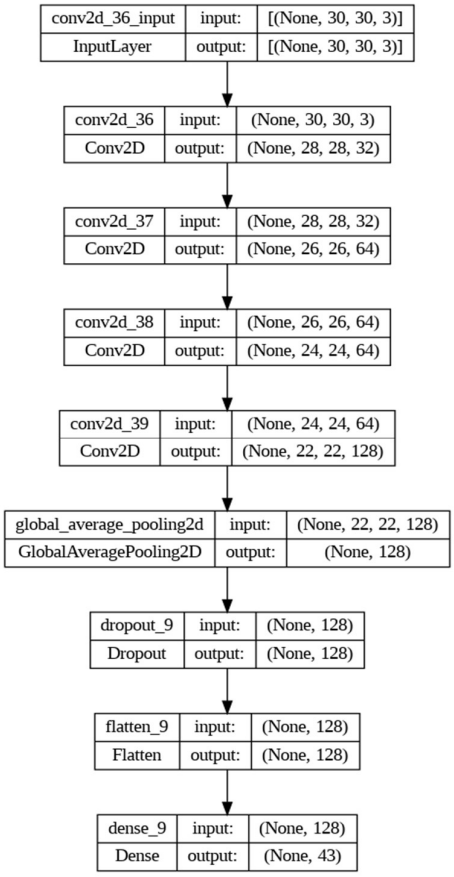
شبکه پیچشی:

مدل خلق شده شامل لایه اول کانولوشن با سایز ۳*۳ و ۳۲ نورون، لایه دوم کانولوشن با همان سایز و ۶۴ نورون، لایه سوم max-pooling، لایه چهارم کانولوشن ۶۴ نورونه ۳*۳، لایه پنج کانولوشن ۳*۳ ۱۲۸ نورونه و لایه بعد max-pooling قرار گرفت. در نهایت هم یک لایه dropout ٪۵۰ و لایه dense قرار گرفت. در این قسمت از بهینه ساز adams استفاده شد. مدل های بعدی به مراتب ضعیف تر عمل کردند لذا در گزارش به این مدل مدل بهینه میگوییم.

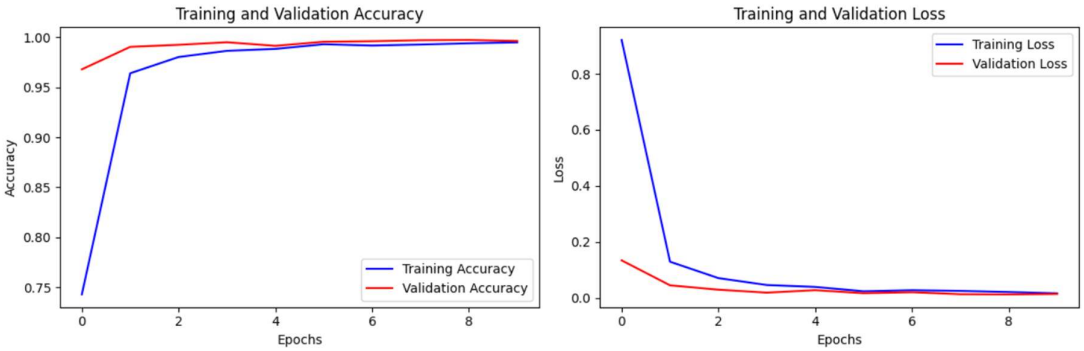
دقت این شبکه برای داده های تست نزدیک به ٪۹۷ بود. دقت و loss در هر ایپاک هم در نمودار رسمش شده.

جدول ۲: دقت شبکه **CNN** بهینه

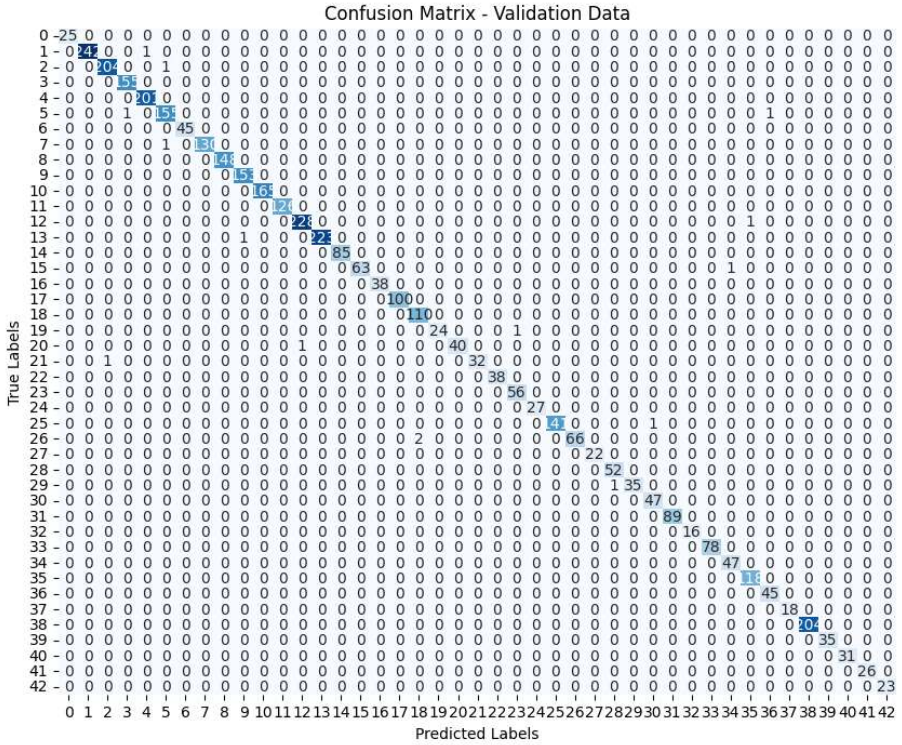
Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9749	0.1275	0.9959	**0.0155**



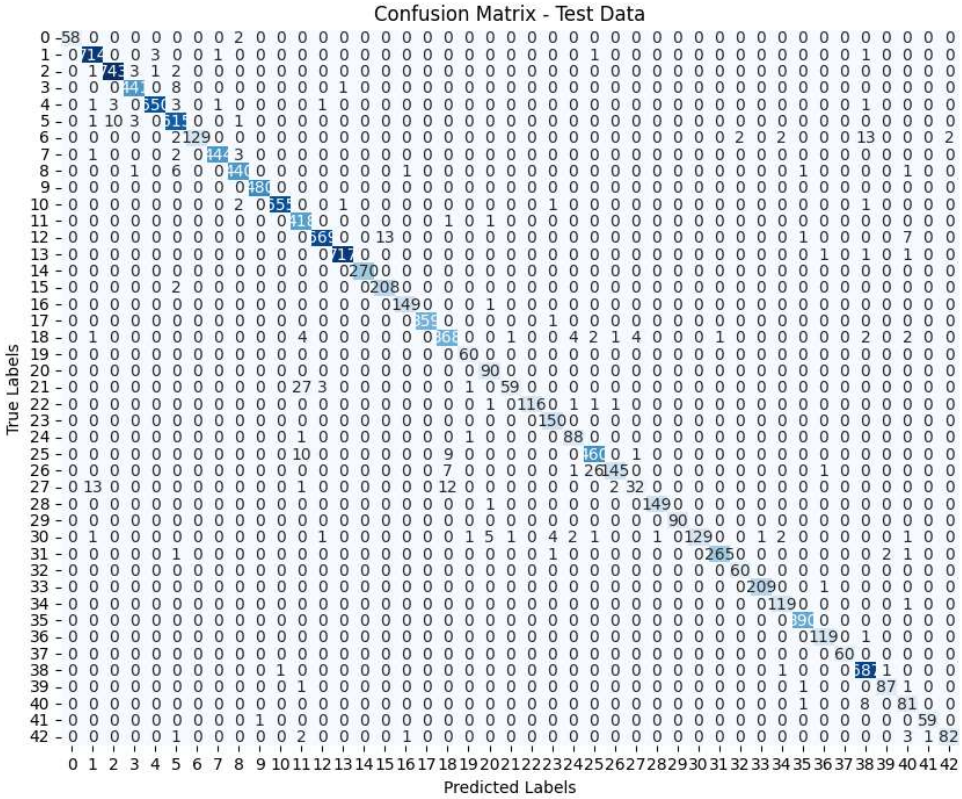
شکل ۸: شبکه پیچشی ۱



شکل ۹: نمودار عملکرد مدل در هر ایپاک



شکل ۱۰: ماتریس سردرگمی داده های **Val** شبکه **CNN** بهینه



شکل ۱۱: ماتریس سردرگمی داده های تست شبکه‌ی **CNN** بهینه

بدون پولینگ:

در این بخش از شبکه اصلی لایه های پولینگ حذف شد. برای جلوگیری از شلوغی گزارش تصاویر شبکه در اینجا به صورت کامل قرار نگرفته. نتایج این مدل در جدول زیر قرار گرفته. با توجه به مقادیر دقت میتوان دید که داده ها با وجود عملکرد خوب بر val عملکرد خوبی بر داده test نداشته اند.

جدول ۳: دقت شبکه **CNN** بدون **pooling**

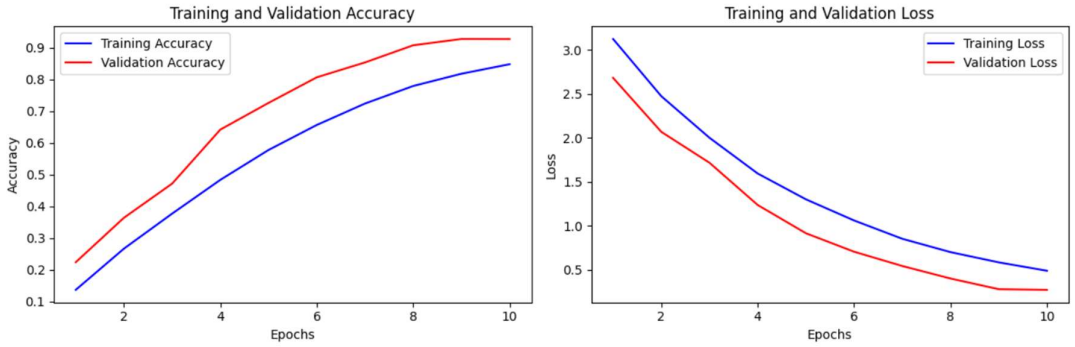
Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9561	0.3792	0.9921	**0.0457**

با average pooling:

نکته جالب توجه این است که نمودار آموزش در این شبکه به نرمی به دقت مطلوب رسیده. اما دقت افت شدیدی داشته.

جدول ۴: دقت شبکه‌ی **CNN** با **average pooling**

Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9069	0.3760	0.9668	**0.1587**



شکل ۱۲: نمودار تغییرات دقت و **Loss** در فرایند آموزش

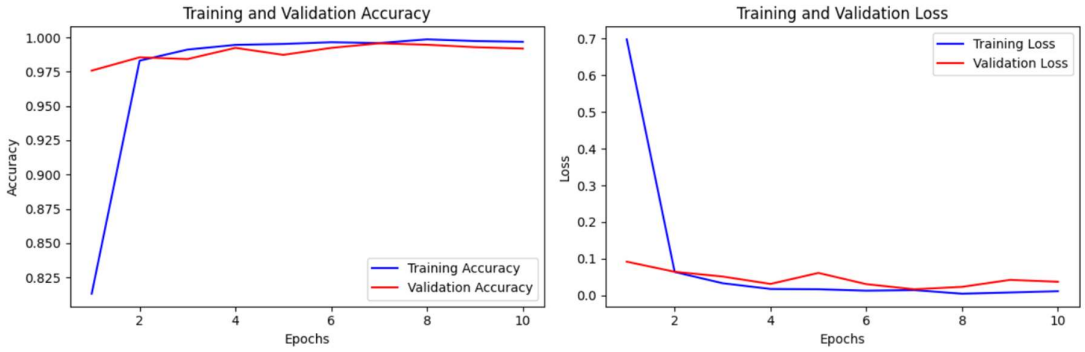
حذف dropout:

در این مدل دقت داده های تست حدود ۱ درصد کاهش یافته.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_52 (Conv2D)          (None, 28, 28, 32)        896

 conv2d_53 (Conv2D)          (None, 26, 26, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 conv2d_54 (Conv2D)          (None, 11, 11, 64)        36928

 conv2d_55 (Conv2D)          (None, 9, 9, 128)         73856

 max_pooling2d_5 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten_13 (Flatten)        (None, 2048)              0

 dense_13 (Dense)            (None, 43)                88107

=================================================================
Total params: 218,283
Trainable params: 218,283
Non-trainable params: 0
```

جدول ۵: دقت شبکه‌ی **CNN** بدون **dropout**

Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9636	0.2796	0.9946	**0.0252**



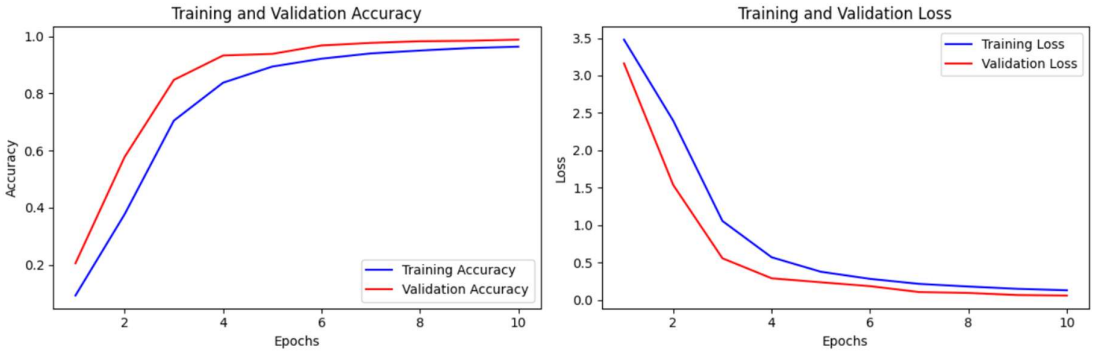
شکل ۱۳: نمودار تغییرات دقت و **Loss** در فرایند آموزش

با gradient decent:

در این روش دقت اندکی کاهش یافت و در نمودار میتوان دید سرعت فیت شدن داده ها کمتر است و نمودارها در هر ایپاک تغییر کمتری دارند. احتمالا با چند ایپام بیشتر این مدل هم به دقت بهتری میرسید.

جدول ۶: دقت شبکه‌ی **CNN** با **gradient decent**

Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9339	0.2829	0.9837	**0.0769**



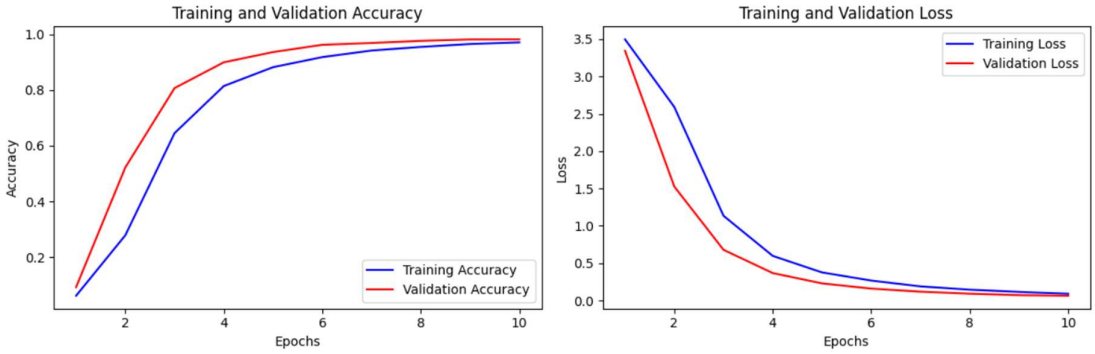
شکل ۱۴:: نمودار تغییرات دقت و **Loss** در فرایند آموزش

Sigmid:

این فعال سازی هم دقت را کاهش داد.

جدول ۷: دقت شبکه‌ی **CNN** با **gradient decent**

Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9479	0.1994	0.9923	**0.0321**



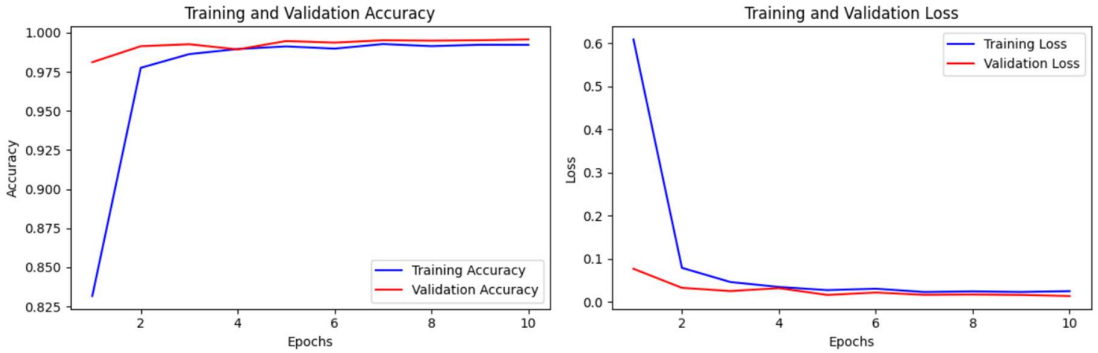
شکل ۱۵:: نمودار تغییرات دقت و **Loss** در فرایند آموزش

Tanh:

جدول ۸: دقت شبکه‌ی **CNN** با **gradient decent**

Test Accuracy	Test Loss	Val Accuracy	Val Loss

0.9711	0.1589	0.9962	**0.0116**



شکل ۱۶:: نمودار تغییرات دقت و **Loss** در فرایند آموزش

ب) Data Augmentation:

از این تکنیک برای افزایش تعداد نمونه های دیتاست استفاده میشود. در این روش به صورت مصنوعی با ایجاد تغییراتی در تصاویر داده های جدید با همان لیبل تولید میکنیم. این روش کمک میکند تا شبکه به صورت جامع تر دادگان را بشناسد. برای مثال با چرخاندن تصاویر تابلو راهنمایی رانندگی شبکه حساسیت خود را به افقی بودن تابلوی ورود ممنوع از دست میدهد و در صورت وجود تصویر کج خطا ایجاد نمیشود. از تکنیک های زیر برای ایجاد دیتا استفاده میشود.

Flipping, Rotation, Scaling, Translation, Shearing, Adding noise, Cropping

با توجه به وجود فلش در داده ها استفاده از flipping و rotation با زاویه زیاد مجاز نیست چون تابلو تغییر میکند. برای مثال تصاویر زیر دو تابلوی متفاوت هستند که ممکن است با تغییرات فوق به یک دیگر تبدیل شوند.

شکل ۱۷: نمونه ای از دسته ها که با **flip** شدن یکسان میشوند.

در ادامه با سه روش scaling، shearing و اضافه کردن نویز که به نظر کارآمد می آیند، داده های train را افزایش داده و شبکه را آموزش دادیم. برای این قسمت از بهترین شبکه پیچشی قسمت ۱ استفاده کردیم. پارامترهای افزون شده به دیتا چند بار تست شد که بهترین آن در شکل زیر آمده است.

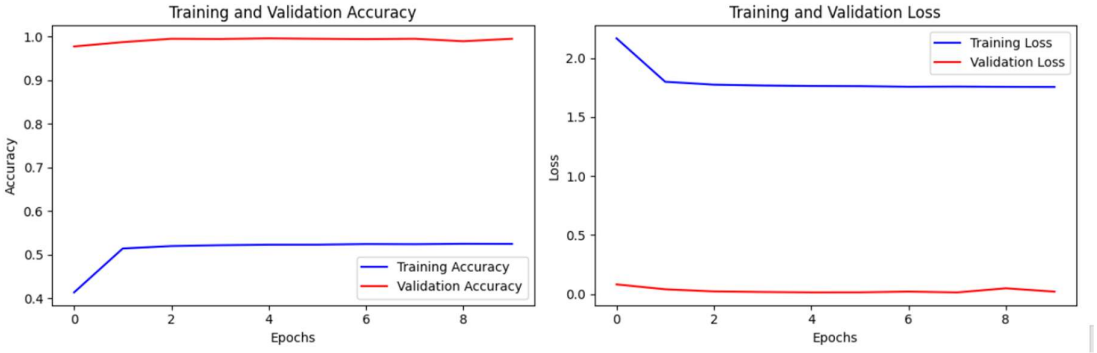
```python
datagen = ImageDataGenerator(
    shear_range=0.2,  # Shearing
    zoom_range=0.05 ,   # Scaling
    rotation_range=10,  # rotation 10 degrees
    brightness_range=[0.5, 1.5],  # Adding noise (brightness)
)
```

شکل ۱۸: هایپرپارامترهای **Data Augmentation**

در این حالت دقت تست به ۰,۹۷۷۸ درصد رسید و افزایش ۰,۴ درصدی نسبت به تربیت با دیتای اورجینال داشت.

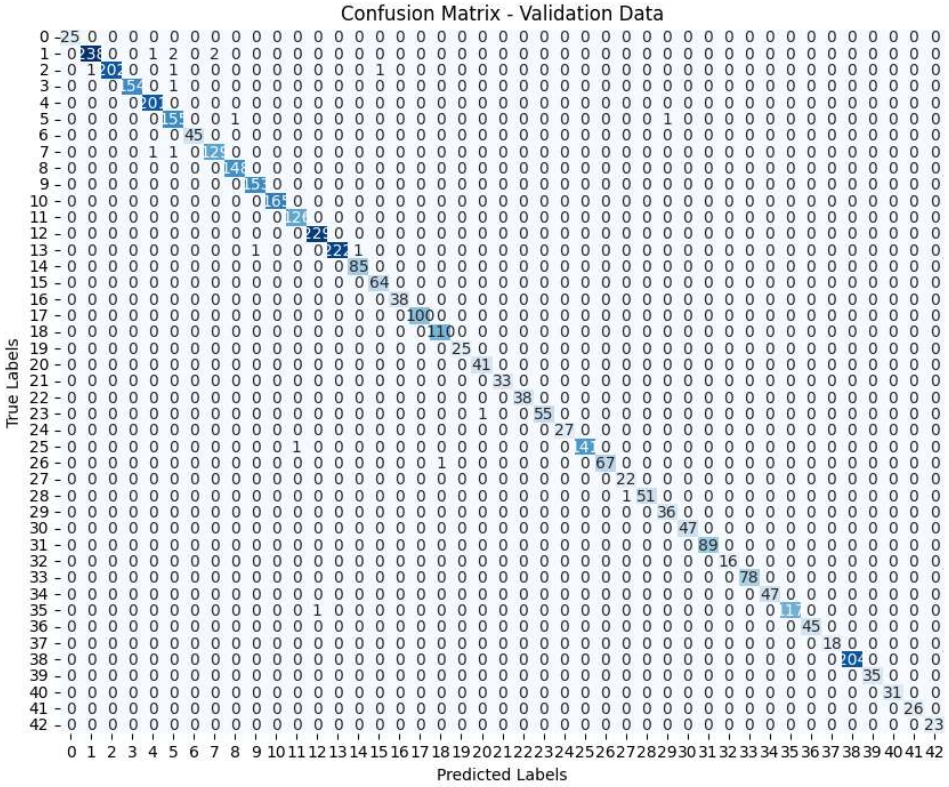
جدول ۹: دقت شبکه‌ی **CNN** بر دیتای افزون شده☺

Test Accuracy	Test Loss	Val Accuracy	Val Loss
0.9778	0.1280	0.9974	**0.0132**

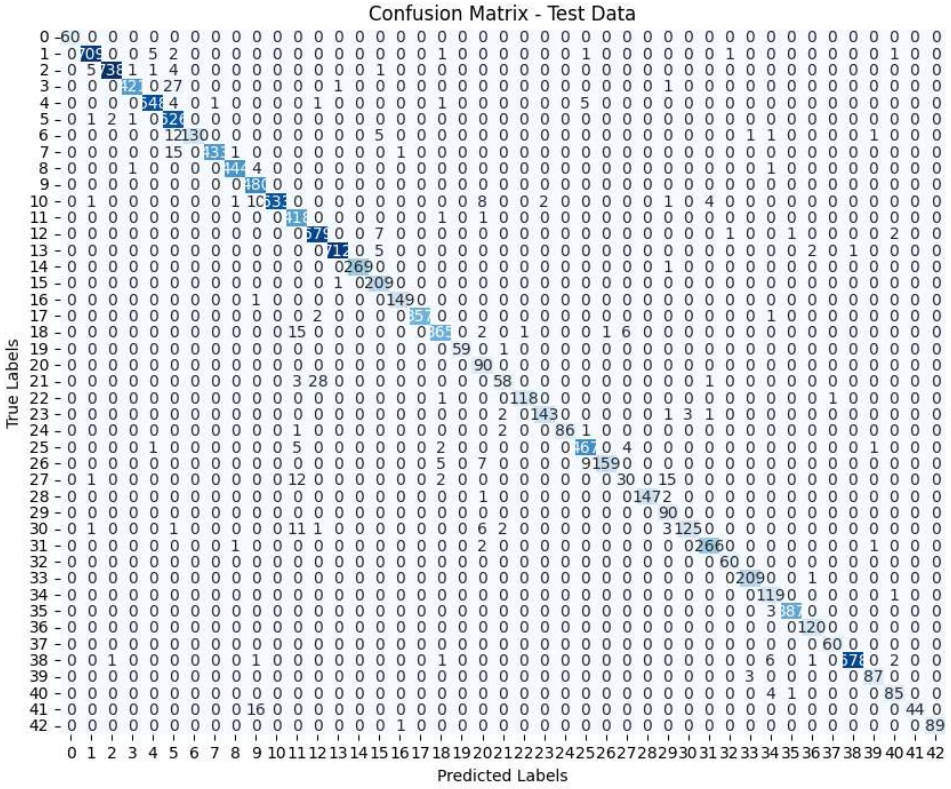


شکل ۱۹ **:**نمودار تغییرات دقت و **Loss** در فرایند آموزش

در نهایت ماتریس سردرگمی برای این مدل ایجاد شد.



شکل ۲۰:ماتریس سردرگمی داده های **Val** شبکه‌ی **CNN** بهینه که با تصاویر افزون آموخته



شکل ۲۱:ماتریس سردرگمی داده های تست شبکه‌ی **CNN** بهینه که با تصاویر افزون آموخته

پ) مقایسه نتایج:

در قسمت الف MLP کارایی متوسطی را برای دسته بندی این دادگان نشان داد، اما در شبکه پیچشی به دقت بالای ۹۷ رسیدیم. بهترین شبکه پیچشی به همراه لایه های max_pooling و dropout بود که نتایج آن در جدول زیر و ستون CNN آمده است. ایجاد تغییرات در دیتای ترین نیز میتواند باعث بهتر شدن یا بدتر شدن دقت شود که با تست چند حالت ایجاد تغییر تصاویر افزون شده مناسبی ساخته شد که باعث بهبود دقت داده های تست شد.

جدول ۱۰: مقایسه نتایج

	Val Loss	Val Accuracy	Test Loss	Test Accuracy
MLP	**۰.۴۰۲۷**	۰.۹۱۰۲	۰.۸۷۲	۰.۸۲۳۱
CNN	0.0155	0.9959	0.1275	0.9749
Augmented CNN	0.0132	0.9974	0.1280	0.9778