به نام خدا

دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

تشخیص و شناسایی خطا

گزارش پروژه پایانی

شیما سادات ناصری

۴۰۱۱۲۸۱۴

دکتر مهدی علیاری شوره دلی

تیر ماه ۱۴۰۲

فهرست مطالب

عنوان شماره صفحه

فهرست اشکال

چکیده

داده‌های مورد استفاده، داده‌های شبیه سازی شده یک بویلر گازی است که تحت شرایط مختلفی بررسی و داده‌برداری شده‌است. در کل داده‌ها دارای ۱ حالت سالم و ۴ خطا بوده و ۶ ویژگی دارد. بر روی این داده‌ها عملیات‌های مختلفی از جمله MLP، CNN، Hybrid network و Transformer بررسی شد. در نهایت هر کدام از این شبکه‌ها براساس معیارهای خروجی بدست آمده آن‌ها، نمودارهای t-SNE آن‌ها مقایسه شدند و مشاهده شد که CNN بهترین پاسخ را دارد.

بخش۱: بررسی داده

بعد از بررسی‌هایی که در سایت‌های مختلف برای یک دیتاست مناسب برای ادامه پروژه انجام شد، دیتاست SIMULATED BOILER DATA FOR FAULT DETECTION AND CLASSIFICATION انتخاب شدند. این داده ها بر اساس یک شبیه‌ساز برای بویلر گاز سوز Viessmann Vitorond 200 VD2 Series 380 در Matlab/Simulink بر اساس مدل بویلر مجازی برپایه Simscape بدست آمده است. پس از تأیید اعتبار با داده های آزمایش سازنده، یک سری خطا (هوای اضافی، رسوب مبدل حرارتی، مقیاس بندی عنصر مبدل حرارتی سمت آب و شرایط احتراقی lean در بویلر) با این شبیه ساز، همراه با شرایط عملیاتی اسمی به عنوان داده نرمال برای طیف وسیعی از پارامترهای عملیاتی (نرخ سوخت گاز از ۱ کیلوگرم بر ثانیه تا ۴ کیلوگرم بر ثانیه، سرعت جریان جرمی آب از ۳ کیلوگرم بر ثانیه تا ۱۲.۵ کیلوگرم بر ثانیه و دمای هوای احتراق از ۲۸۳ کلوین تا ۳۰۳ کلوین) ارائه شده است. از روش نمونه گیری فاکتوریل استفاده شد که در مجموع ۲۷۲۸۱ شبیه سازی انجام شد. این مجموعه داده شامل نقاط قابل مشاهده برای یک سیستم اتوماسیون ساختمان به همراه Condition (کلاس تفصیلی) و Class (برچسب) است.

مشکل مهمی که این داده برای ما در این قسمت دارد این است که دارای داده‌های غیر عددی یا اسمی است. این داده‌ها در دو بخش تفصیلی و برچسب هستند. کلیات آن‌ها به صورت زیر می‌باشد.

```
1  data=pd.read_csv('data/Boiler_emulator_dataset.txt',delimiter=",")
2  data.head()
```
✓ 0.0s

	Fuel_Mdot	Tair	Treturn	Tsupply	Water_Mdot	Condition	Class
0	1	283	333.0	363.574744	3.0	%=0.05	Lean
1	1	283	333.0	362.349517	3.0	%=0.1	Nominal
2	1	283	333.0	361.216941	3.0	%=0.15	ExcessAir
3	1	283	333.0	360.166890	3.0	%=0.20	ExcessAir
4	1	283	333.0	359.190662	3.0	%=0.25	ExcessAir

```
1  pd.unique(data['Class'])
```
✓ 0.0s

```
array(['Lean', 'Nominal', 'ExcessAir', 'Fouling', 'Scaling'], dtype=object)
```

```
1  pd.unique(data['Condition'])
```
✓ 0.0s

```
array(['%=0.05', '%=0.1', '%=0.15', '%=0.20', '%=0.25', '%=0.3', '%=0.35',
       '%=0.40', '%=0.45', '%=0.50', 'F = 0.01', 'F = 0.06', 'F = 0.11',
       'F = 0.16', 'F = 0.21', 'F = 0.26', 'F = 0.31', 'F = 0.36',
       'F = 0.41', 'F = 0.46', 'S = 0.01', 'S = 0.06', 'S = 0.11',
       'S = 0.16', 'S = 0.21', 'S = 0.26', 'S = 0.31', 'S = 0.36',
       'S = 0.41', 'S = 0.46', 'Nominal'], dtype=object)
```

شکل ۱- کد اصلاح ساختار داده

مشاهده می‌شود که تمامی داده‌ها در این دو گروه، از جنس عدد نیستند؛ درنتیجه نیازمند جایگزینی با اعداد مناسب هستند. برای هر داده در گروه تفصیلی می‌توان عدد مقابل آن‌را به عنوان یک ویژگی برای اعلام اهمیت داده استفاده کرد. برای گروه کلاس هم هر کلاس را با کمک دیکشنری به ۰، ۱، ۲، ۳ و ۴ اختصاص داد. کد مربوطه به صورت زیر می‌باشد.

```
replacements = {'Nominal': 0, 'Lean': 1, 'ExcessAir': 2, 'Fouling': 3,'Scaling' : 4,'%=0.05' : 0.05, '%=0.1': 0.1, '%=0.15': 0.15,
                '%=0.20': 0.2, '%=0.25' : 0.25, '%=0.3': 0.3, '%=0.35': 0.35, '%=0.40': 0.4,  '%=0.45': 0.45, '%=0.50': 0.5, 'F = 0.01': 0.01,
                'F = 0.06': 0.06, 'F = 0.11': 0.11, 'F = 0.16': 0.16, 'F = 0.21': 0.21, 'F = 0.26': 0.26, 'F = 0.31': 0.31, 'F = 0.36': 0.36,
                'F = 0.41': 0.41, 'F = 0.46': 0.46, 'S = 0.01': 0.01,'S = 0.06': 0.06, 'S = 0.11': 0.11, 'S = 0.16': 0.16, 'S = 0.21': 0.21,
                'S = 0.26': 0.26, 'S = 0.31': 0.31, 'S = 0.36': 0.36,'S = 0.41': 0.41, 'S = 0.46': 0.46 }

data=data.replace(replacements)
data.to_csv('data.csv', index=False)
```

شکل ۲- داده‌هایی که نیاز به جایگزینی دارند

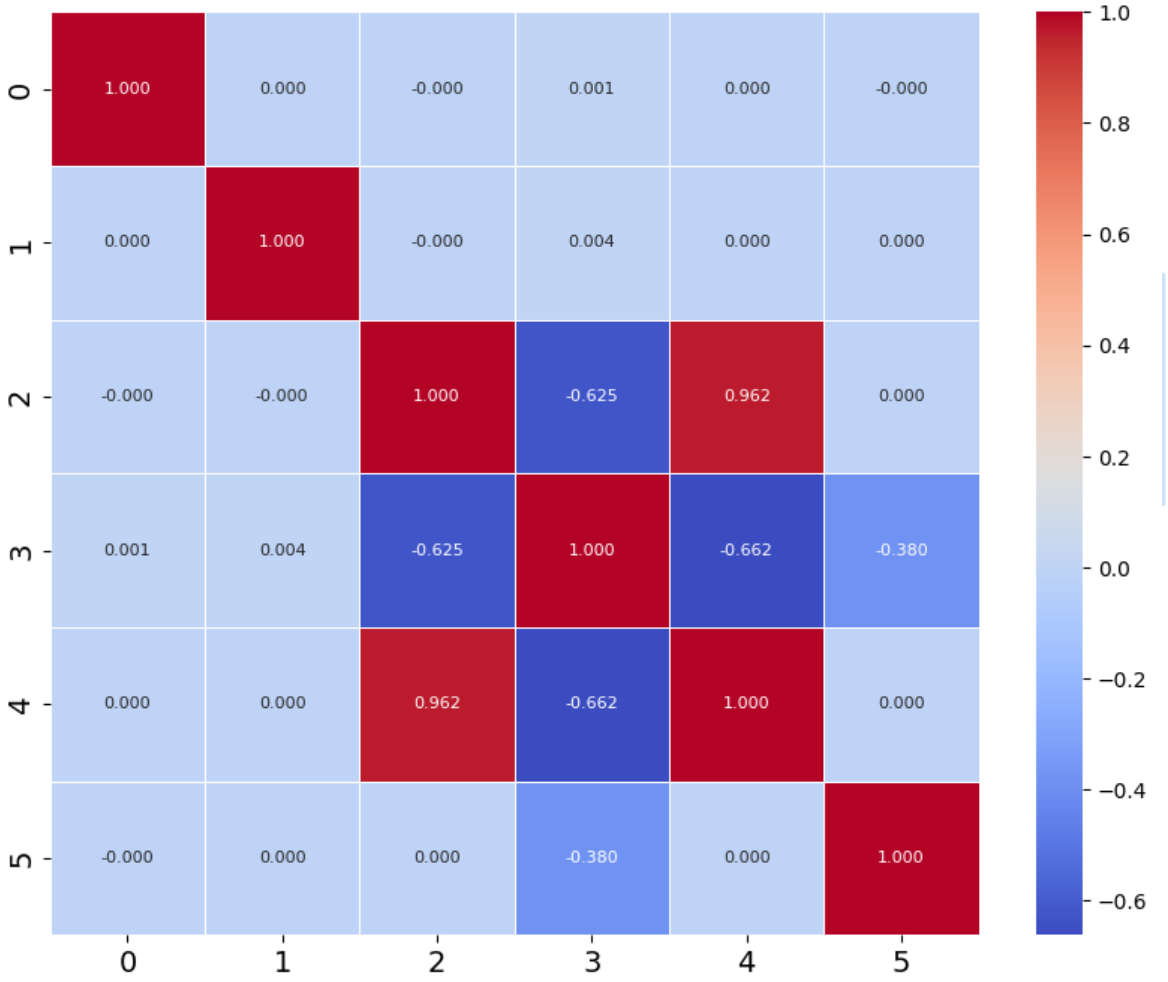
داده جدید به صورت زیر تبدیل می‌شود.

Fuel_Mdot	Tair	Treturn	Tsupply	Water_Mdot	Condition	Class
1	283	333	363.574744	3	0.05	1
1	283	333	362.3495168	3	0.1	0
1	283	333	361.216941	3	0.15	2
1	283	333	360.1668904	3	0.2	2
1	283	333	359.1906622	3	0.25	2

شکل ۳- نمایی از شکل نهایی داده

در کل ۶ ویژگی برای این داده در نظر گرفته خواهد شد که به جز ستون کلاس در جدول بالا دیده می‌شود. نمودار correlation این داده‌ها به صورت زیر می‌باشد.

جدول ۱- ماتریس **Correlation** ویژگی‌های داده



مشکل بزرگی که در این داده وجود دارد این است که ویژگی‌های T_return و T_supply و Water_Mdot نسبت به هم دارای correlation منفی هستند و این می‌تواند در امر کاهش بُعد مشکل ساز شود.

بخش ۲: بررسی اثر کاهش بعد بر روی داده‌ها

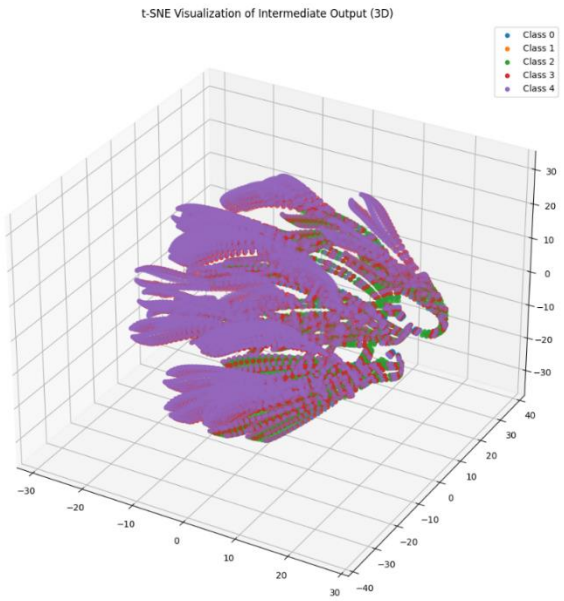
از مهم‌ترین مرحله‌ها در هر بررسی داده، کاهش بعد برای افزایش سرعت محاسبات است. البته ممکن است موجب از دست رفتن برخی از داده‌های مفید شود اما کمک زیادی در طبقه بندی داده‌ها می‌کند. داده‌ی مورد استفاده در این پروژه دارای ۶ ویژگی قابل آموزش برای شبکه است که می‌توانند کاهش یابند. در ادامه چند روش مهم و پرکاربرد برای کاهش بعد را روی این داده بررسی می‌کنیم:

روش t-SNE

روش t-SNE (t-Distributed Stochastic Neighbor Embedding) یک روش محبوب کاهش بعد غیرخطی است که برای نمایش داده‌های فضاهای بُعدی بالا در یک فضای کمتر بعد استفاده می‌شود. این تکنیک به ویژه برای آشکارسازی خوشه‌ها و الگوها در داده‌ها بسیار مؤثر است.

t-SNE در سال ۲۰۰۸ توسط لورنس فان دِر ماتِن (Laurens van der Maaten) و جفری هینتون (Geoffrey Hinton) معرفی شد. این تکنیک بر پایه مفهوم Stochastic Neighbor Embedding (SNE) استوار است اما برخی از محدودیت‌های آن را برطرف می‌کند. t-SNE یک توزیع احتمالی بر روی جفت‌های اشیاء بُعد بالا و یک توزیع مشابه بر روی جفت‌های متناظر آن‌ها در بُعد کمتر بنا می‌کند. هدف آن کاهش اختلاف بین این دو توزیع را با بهینه‌سازی تعبیه‌گر است.

خروجی داده بدست آمده به صورت زیر است.

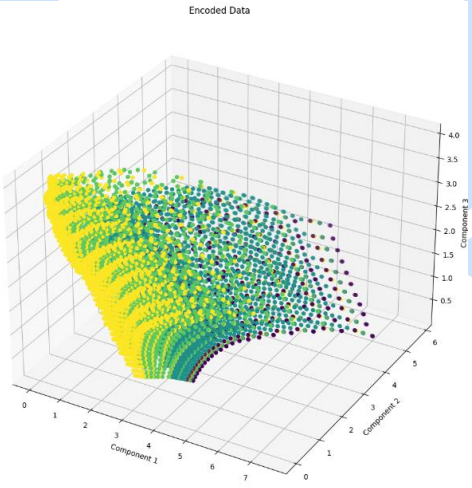


شکل ۴- خروجی **t-SNE** داده خام

روش Autoencoder

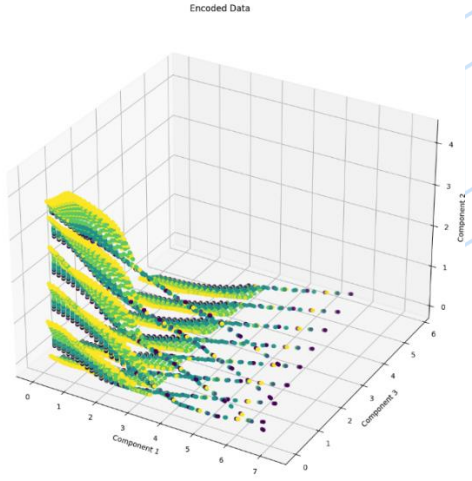
اتوانکدر (Autoencoder) نوعی شبکه عصبی مصنوعی است که برای یادگیری بدون ناظر و کاهش بُعد استفاده می‌شود. این شبکه شامل یک شبکه کدگذار است که داده ورودی را به یک نمایش کم‌بُعدی، که "کُدگذاری" نامیده می‌شود، نگاشت می‌دهد و یک شبکه بازگشتی است که سعی در بازسازی داده ورودی از کُدگذاری دارد. هدف اتوانکدر کمینه کردن خطای بازسازی است که مدل را به یادگیری نماینده‌های معنادار از داده ورودی تشویق می‌کند. برای بررسی بیشتر این روش روی داده‌ها، از دو حالت ۳ ویژگی و ۴ ویژگی در خروجی به صورت جداگانه خروجی گرفته شده‌است.

نمایش داده‌های انکد شده برای ۳ ویژگی به صورت زیر می‌باشد.



شکل ۵- نمایی از خروجی اتوانکدر برای ۳ ویژگی

برای ۴ ویژگی نیز به صورت زیر بدست آمده که در اینجا ۳ ویژگی اول نمایش داده شده‌است.

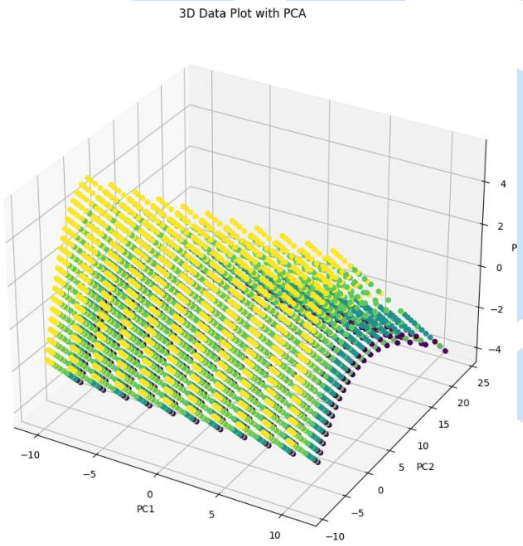


شکل ۶- نمایی از خروجی اتوانکدر برای ۴ ویژگی (۳ ویژگی نمایش داده می‌شوند)

روش PCA

PCA (تحلیل مؤلفه‌های اصلی) یکی از محبوب‌ترین تکنیک‌های کاهش بُعد خطی است که برای تبدیل داده‌های فضاهای بُعدی بالا به یک فضای بُعد کمتر با حفظ اطلاعات مهم استفاده می‌شود. این تکنیک با شناسایی مؤلفه‌های اصلی، که جهت‌های عمودی‌ای هستند و بیشترین واریانس را در داده‌ها ثبت می‌کنند، این کاهش بُعد را انجام می‌دهد.

خروجی بدست آمده از این روش به صورت زیر می‌باشد.

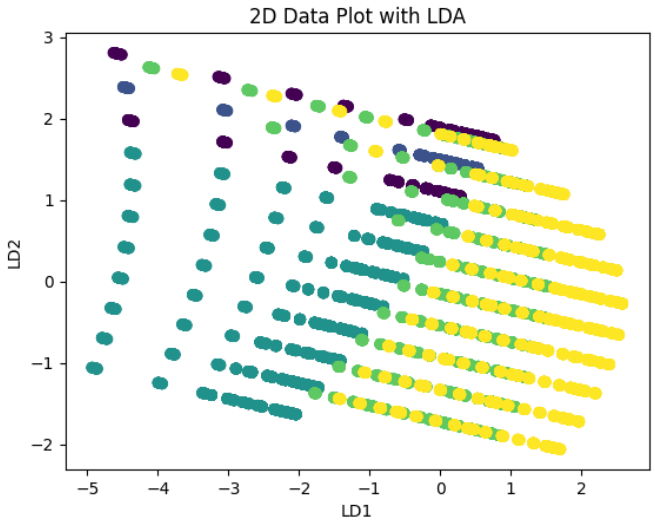


شکل ۷- نمایی از خروجی **PCA**

روش LDA

LDA (تحلیل ممیز خطی) یک تکنیک کاهش بعد است که هدف آن پیدا کردن ترکیب خطی از ویژگی‌ها است که بیشینه جدایی بین کلاس‌ها در داده را فراهم می‌کند. این تکنیک به طور معمول در وظایف طبقه‌بندی استفاده می‌شود تا داده‌ها را روی یک فضای کم‌بعدی نگاشت کند و هم‌گسیلی کلاس‌ها را بیشینه سازد.

به دلیل محدودیت‌هایی که داده نسبت به این الگوریتم داشت، تنها ۲ ویژگی قابل استخراج بود. خروجی این روش به صورت زیر است.



شکل ۸- نمایی از خروجی **LDA**

تنها موردی که باعث نگرانی است این است که داده‌های زیادی ممکن است در کاهش بعد از ۶ ویژگی به ۲ ویژگی رخ دهد.

مقایسه روش‌های مختلف

برای بررسی بهتر داده های بدست آمده از هر روش معیارهای Silhouette Score، Calinski-Harabasz Index و Davies-Bouldin Index استفاده می‌کنیم. تحلیل هر معیار به صورت زیر می‌باشد.

معیار Silhouette Score اندازه‌گیری می‌کند که هر نمونه در یک مجموعه داده در مقایسه با خوشه‌های دیگر چقدر در خوشه اختصاص داده شده خود قرار می‌گیرد. از ۱- تا ۱ متغیر است، جایی که مقدار بالاتر نشان دهنده عملکرد بهتر خوشه بندی است. نمره نزدیک به ۱ نشان می‌دهد که نمونه‌ها به خوبی خوشه‌بندی شده‌اند، در حالی که نمرات نزدیک به ۰ نشان‌دهنده همپوشانی خوشه‌ها و نمرات منفی نشان می‌دهد که ممکن است نمونه‌ها به خوشه اشتباهی اختصاص داده شده باشند. امتیاز Silhouette را می توان برای ارزیابی فشرده بودن و جداسازی خوشه ها، ارائه بینشی در مورد کیفیت بازسازی داده ها یا نتایج خوشه بندی استفاده کرد.

شاخص Calinski-Harabasz که به عنوان معیار نسبت واریانس نیز شناخته می‌شود، اندازه‌گیری نسبت بین پراکندگی درون خوشه‌ای و پراکندگی بین خوشه‌ای است. هدف آن به حداکثر رساندن نسبت است که نشان دهنده خوشه‌های به خوبی جدا شده و فشرده است. مقدار بالاتر Calinski-Harabasz Index مربوط به عملکرد خوشه بندی بهتر است. از این شاخص می توان برای ارزیابی کیفیت تخصیص خوشه ها و فشرده بودن داده های بازسازی شده استفاده کرد.

شاخص Davies-Bouldin شباهت بین خوشه ها را با در نظر گرفتن پراکندگی درون خوشه ای و جدایی بین خوشه ای ارزیابی می کند. میانگین شباهت بین هر خوشه و مشابه ترین خوشه را با در نظر گرفتن اندازه آنها اندازه گیری می کند. مقدار پایین‌تر Davies-Bouldin Index نشان‌دهنده عملکرد بهتر خوشه‌بندی است، با مقادیر کوچک‌تر نشان‌دهنده خوشه‌های فشرده‌تر و متمایزتر است. این شاخص می تواند به ارزیابی کیفیت نتایج خوشه بندی یا بازسازی داده ها با در نظر گرفتن تعادل بین فاصله های درون خوشه ای و بین خوشه ای کمک کند.

مقادیر معیارهای توضیح داده شده در بالا به صورت زیر می‌باشد.

جدول ۲- معیارهای محاسبه شده برای هر روش کاهش بعد

	Silhouette Score	Calinski-Harabasz Index	Davies-Bouldin Index
Main data	-0.02545	647.56	242.78
t-sne	-0.04240	266.04	258.81
AE with 3 features	-0.03420	1169.97	199.24
AE with 4 features	-0.01729	66.19	251.75
PCA	-0.02565	657.97	240.34
LDA	0.052537	5271.18	69.30

مشاهده می‌شود که به دلیل پیچیدگی بالای داده، معیار اول خود داده کمتر از صفر شده است. با اینحال، مقدار معیار دوم داده کم نیست و نشان‌گر وجود خوشه‌هایی قابل تمایز از دید واریانسی در داده است. البته در معیار سوم داده اصلی کمی ضعف نسبت به تمایز خوشه‌ها از دید میانگین شباهت بین داده‌ایست.

در کل روش LDA از بین سایر روش‌ها بهتر عمل کرده است. سایر روش‌ها تفاوت چندانی با داده اصلی نداشته و یا حتی داده بدتری را ایجاد کرده‌اند. از طرفی، روش AE قابلیت تکرارپذیری بسیار پایینی دارد و نمی‌توان به خروجی آن اطمینان کرد.

دیدیم که روش‌های کاهش بعد چگونه نسبت به داده‌ی اصلی عمل می‌کنند. در ادامه این پروژه سعی می‌شود تا طبقه بندی کننده‌ی مناسبی برای این داده انتخاب شود.

بخش ۳: روش‌های طبقه بندی

روش‌های طبقه بندی متنوعی برای این دسته از داده‌ها وجود دارد. ابتدا با ساده‌ترین روش‌ها سعی می‌کنیم داده را طبقه بندی کنیم. در ادامه به مرور سعی می‌شود تا شبکه‌های پیچیده‌تری روی داده اعمال شود. دقت شود که بجز قسمت‌هایی که ذکر می‌شود، تمامی روش‌ها بر روی داده اصلی و بدون کاهش بعد انجام شده است.

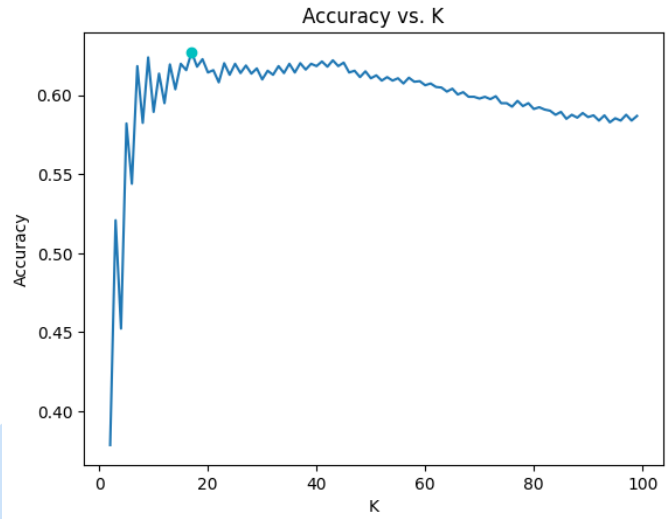
روش KNN

این روش پیچیدگی خاصی ندارد. کد آن به صورت زیر می‌باشد.

```python
from collections import Counter
import numpy as np
from scipy.spatial import cKDTree

class KNNClassifier:
    def __init__(self, k):
        self.k = k

    def fit(self, X, y):
        self.X_train = X
        self.y_train = y
        self.tree = cKDTree(X)

    def predict(self, X):
        _, indices = self.tree.query(X, k=self.k)
        k_labels = self.y_train[indices]
        y_pred = np.array([Counter(labels).most_common()[0][0] for labels in k_labels])
        return y_pred
```

شکل ۹- کد مربوط به روش **KNN**

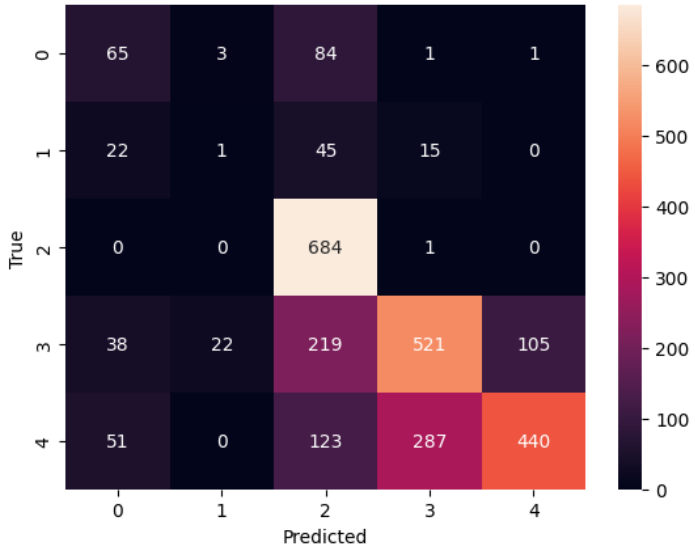
در ادامه سعی می‌شود در بازه ۲ تا ۱۰۰ بهترین k برای داده‌ها پیدا شود. نمودار بدست آمده به ازای مقدار accuracy در هر k به صورت زیر می‌باشد.



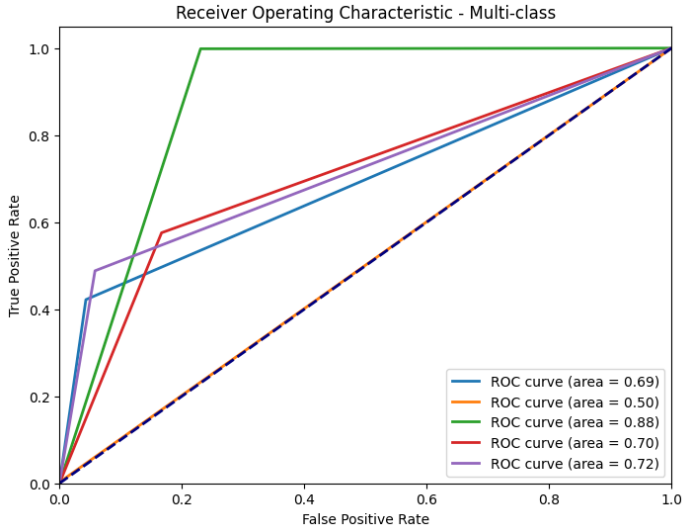
شکل ۱۰-نمودار بدست آمده به ازای مقدار **accuracy** در هر **k** برای ورودی داده خام

بهترین مقدار k برای داده‌ها، ۱۷ بدست آمده است. ماتریس درهم ریختگی آن به صورت زیر می‌باشد.

جدول ۳- ماتریس درهم ریختگی برای **k=17** برای ورودی داده خام



نمودار ROC آن نیز به صورت زیر می‌باشد.



شکل ۱۱-نمودار **ROC** برای **k=17** برای ورودی داده خام

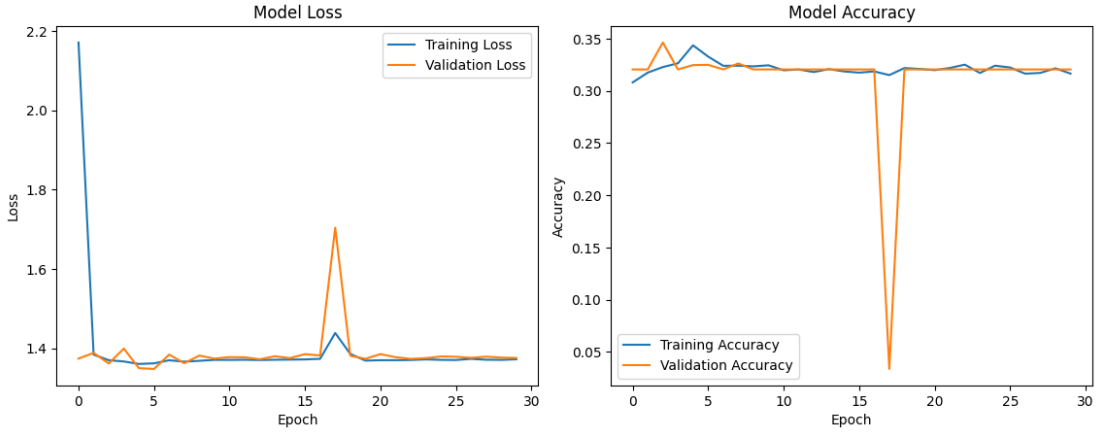
و مقادیر زیر نیز بدست آمده اند.

جدول ۴- معیارهای بدست آمده برای **KNN** و ورودی داده خام

AUC	0.69
Recall	0.62
F1	0.62
Precision	0.62

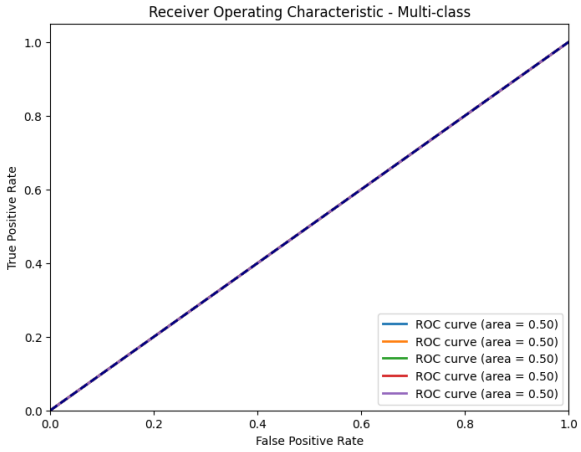
روش MLP

از رایج‌ترین روش‌های iterative می‌باشد. داده‌ها ابتدا از لایه اول با ابعاد (6,1) و تعداد ۳۲ نرون و تابع فعالساز relu عبور می‌کنند. لایه‌های مخفی به ترتیب دارای ۱۶ و ۸ نرون بوده و تابع فعالساز هر دو elu می‌باشد. لایه خروجی نیز به تعداد کلاس‌ها (۵) نرون دارد و تابع فعالساز آن به تبع طبقه بندی کننده بودن شبکه، softmax می‌باشد. از بهینه ساز ADAM با نرخ آموزشی ۰.۰۱ استفاده شده و تابع هزینه آن Sparse Categorical Crossentropy می‌باشد. نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر می‌باشد.



شکل ۱۲- نمودارهای هزینه و دقت برای شبکه **MLP** با ورودی داده خام

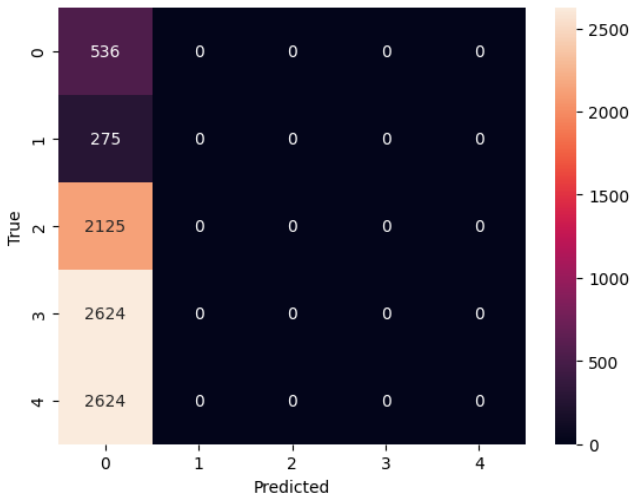
نمودار ROC آن نیز به صورت زیر می‌باشد.



شکل ۱۳- نمودار **ROC** برای شبکه **MLP** با ورودی داده خام

ماتریس درهم ریختگی نیز به صورت زیر می‌باشد.

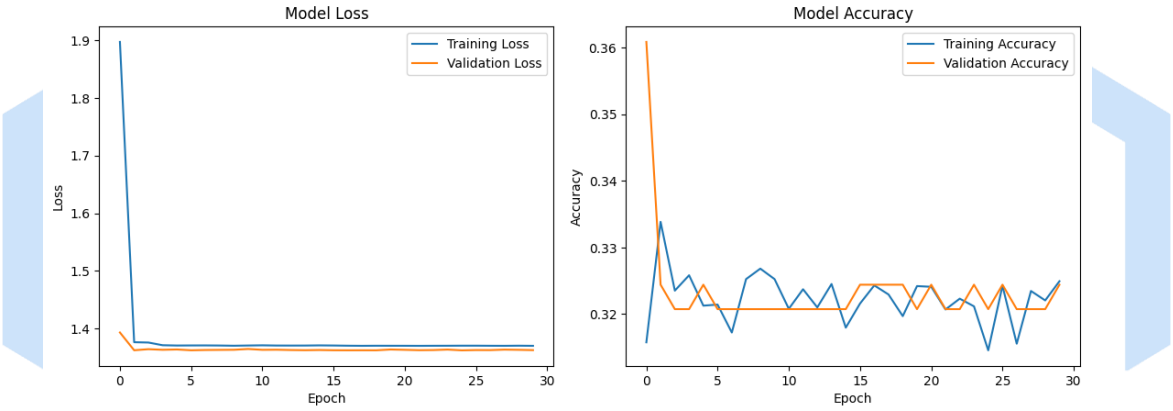
جدول ۴- ماتریس درهم ریختگی برای شبکه **MLP** با ورودی داده خام



مشخصا هیچ داده‌ای را نتوانسته طبقه بندی کند. به همین دلیل سراغ روش‌های دیگر می‌رویم.

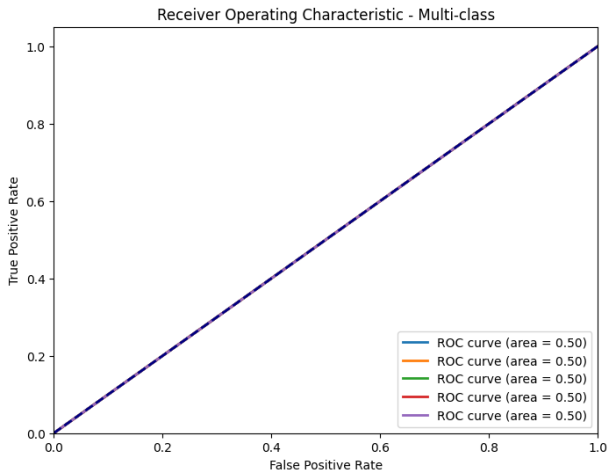
روش CNN

این روش نیز جزو روش‌های رایج در طبقه بندی به خصوص برای تصاویر می‌باشد. لایه اول، یک Conv1D است و دارای ۶۴ فیلتر با اندازه کرنل ۳ و تابع فعالساز relu می‌باشد. یک MaxPooling1D با اندازه پولینگ ۲ پس از آن قرار گرفته است و پس از آن یک لایه Conv1D دیگر با ۳۲ فیلتر با اندازه کرنل ۲ و تابع فعالساز relu وجود دارد. پس از آن، یک لایه Flatten() برای وارد شدن داده به لایه های نرونی ساده می‌شود که این لایه دارای ۱۶ نرون با تابع فعالساز relu می‌باشد. لایه خروجی نیز به تعداد کلاس‌ها (۵) نرون دارد و تابع فعالساز آن به تبع طبقه بندی کننده بودن شبکه، softmax می‌باشد. از بهینه ساز ADAM با نرخ آموزشی ۰.۰۶ استفاده شده و تابع هزینه آن Sparse Categorical Crossentropy می‌باشد. نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر می‌باشد.



شکل ۱۴-نمودار هزینه و دقت برای شبکه **CNN** با ورودی داده خام

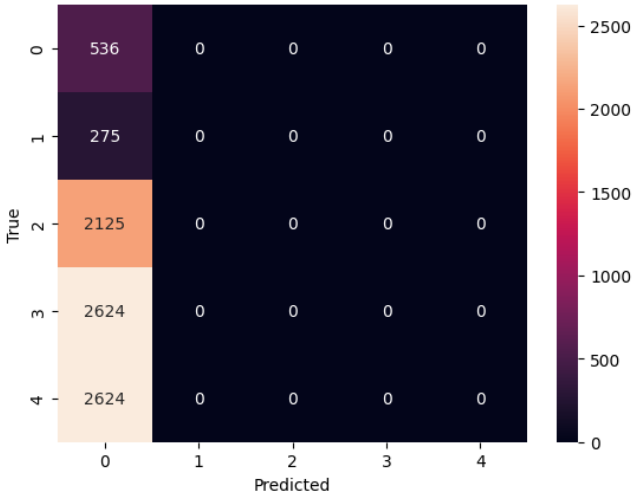
نمودار ROC آن نیز به صورت زیر می‌باشد.



شکل ۱۵-نمودار **ROC** برای شبکه **CNN** با ورودی داده خام

ماتریس درهم ریختگی نیز به صورت زیر می‌باشد.

جدول ۵- ماتریس درهم ریختگی برای شبکه **CNN** با ورودی داده خام

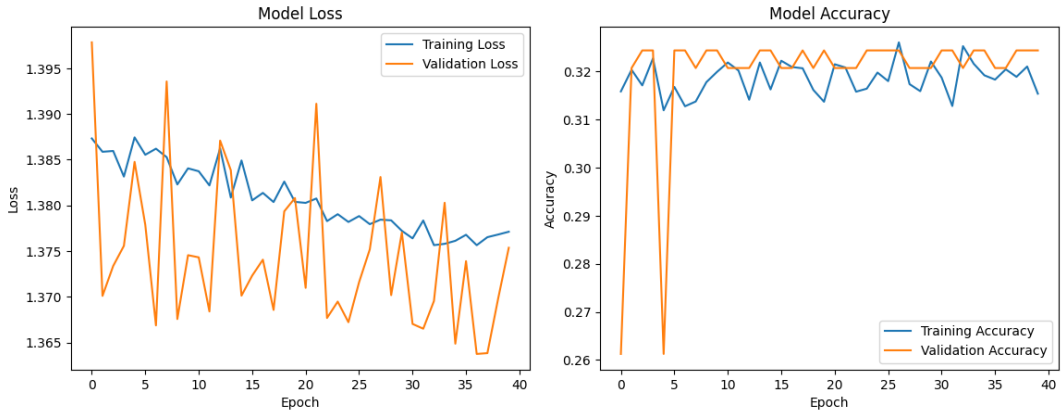


مشخصا بازهم هیچ داده‌ای را نتوانسته طبقه بندی کند. به همین دلیل سراغ روش‌های پیچیده‌تر می‌رویم.

روش هیبرید (LSTM+CNN)

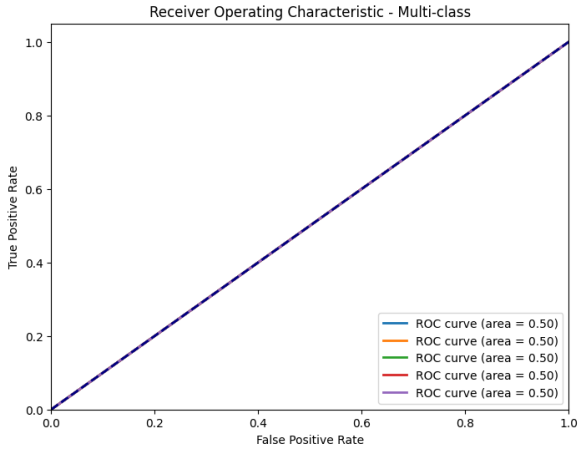
شبکه عصبی مرتبط با کانولوشن (CNN) برای استخراج ویژگی‌های فضایی اس و شبکه عصبی طول کوتاه مدت (LSTM) برای پردازش داده‌های مرتبه‌ای و ثبت الگوها و وابستگی‌های مرتبه‌ای. استفاده CNN در ابتدا و پس از آن LSTM، باعث می‌شود تا اطلاعات فضایی و مرتبه‌ای مدیریت شود و برای وظایف مبتنی بر داده‌های توالی و زمانی کارآمد باشد.

لایه اول این شبکه با ورودی مشابه شبکه‌های قبلی، یک Conv1D با ۳۲ فیلتر، با اندازه کرنل ۳ و تابع فعالساز relu می‌باشد. پس از آن یک MaxPooling1D با اندازه پولینگ ۲ قرار داده شده‌است. لایه بعدی، LSTM با ۱۶ واحد می‌باشد. لایه خروجی نیز مشابه شبکه‌های قبلی درنظر گرفته شده‌است. تابع هزینه و تابع بهینه ساز مشابه توابع قبلی و با نرخ آموزشی ۰.۰۴ در نظر گرفته شده‌است. نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر می‌باشد.



شکل ۱۶- نمودار هزینه و دقت برای شبکه **Hybrid** با ورودی داده خام

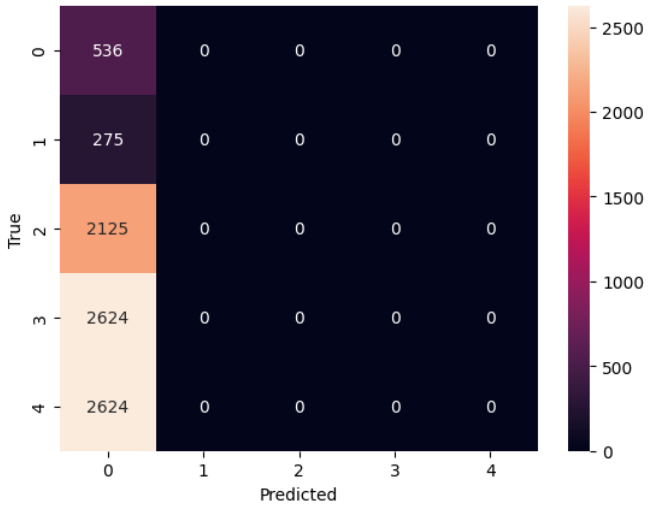
نمودار ROC آن نیز به صورت زیر می‌باشد.



شکل ۱۷- نمودار **ROC** برای شبکه **Hybrid** با ورودی داده خام

ماتریس درهم ریختگی نیز به صورت زیر می‌باشد.

جدول ۶- ماتریس درهم ریختگی برای شبکه **Hybrid** با ورودی داده خام



بازهم هیچ داده‌ای را نتوانسته طبقه بندی کند. از آن‌جایی که این شبکه به اندازه کافی برای داده‌ی ما پیچیده است؛ می‌توان نتیجه گرفت که نیاز به اضافه کردن یکسری پیش پردازش‌ها به داده می‌باشد.

اعمال scaler

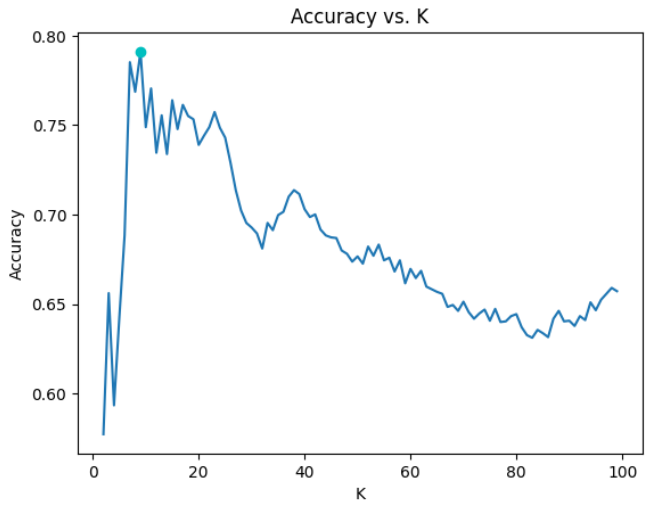
یکی از روش‌های متداول برای پیش پردازش کردن داده‌ها، استفاده از scaler هاست که انواع مختلفی دارند. من در این پروژه از MinMaxScaler استفاده کردم تا تاثیر آن را روی روش‌های قبلی بررسی کنم. برای الگوریتم‌هایی که شامل به‌روزرسانی‌های تکراری هستند (مانند مدل‌های یادگیری عمیق)، داشتن مقیاس‌های یکنواخت در فضای ویژگی، بهبود همگرایی و پایداری آموزش را بهبود می‌بخشد. از طرفی این روش مانند gaussian scaler به نوع توزیع داده‌ها آسیبی وارد نمی‌کند. کد این روش به صورت زیر اعمال شده‌است.

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

شکل ۱۸- کد مربوط به روش **MinMaxScaler**

KNN

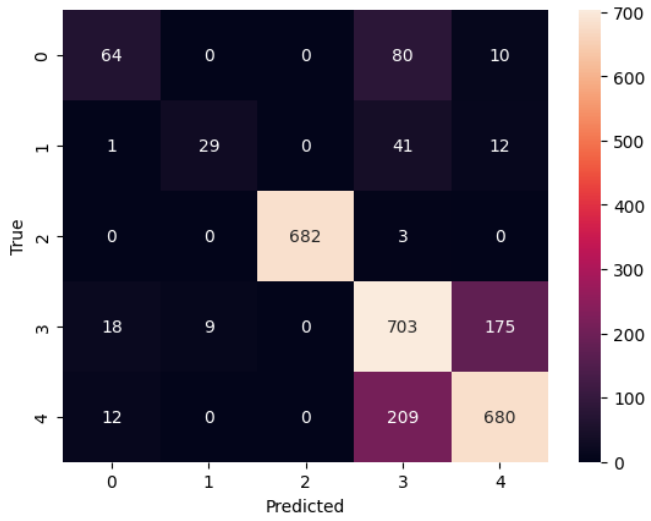
طبقه بندی کننده بعد از اعمال این scaler اجرا شد تا بهترین مقدار k برای داده‌ها پیدا شود. نمودار بدست آمده برای مقدار accuracy در هر k به صورت زیر می‌باشد.



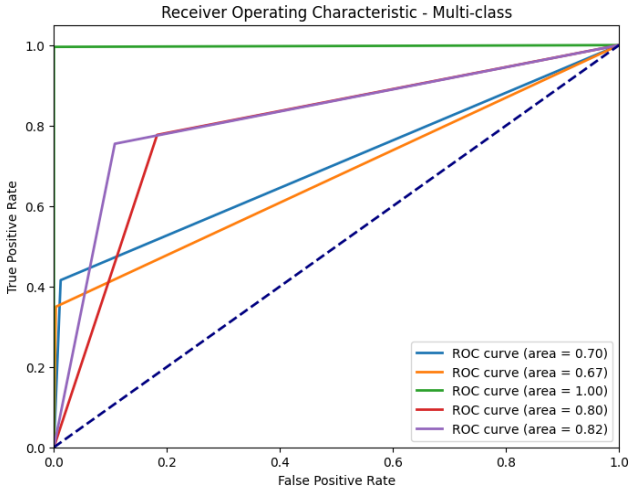
شکل ۱۹- نمودار بدست آمده به ازای مقدار **accuracy** در هر **k** برای ورودی داده **scale** شده

بهترین مقدار k برابر 9 بدست آمد. ماتریس درهم ریختگی آن به صورت زیر می‌باشد.

جدول ۷- ماتریس درهم ریختگی برای **k=9** برای ورودی داده **scale** شده



و نمودار ROC نیز به صورت زیر بدست آمده‌است.



شکل ۲۰- نمودار **ROC** برای **k=9** برای ورودی داده **scale** شده

و مقادیر زیر نیز بدست آمده‌اند.

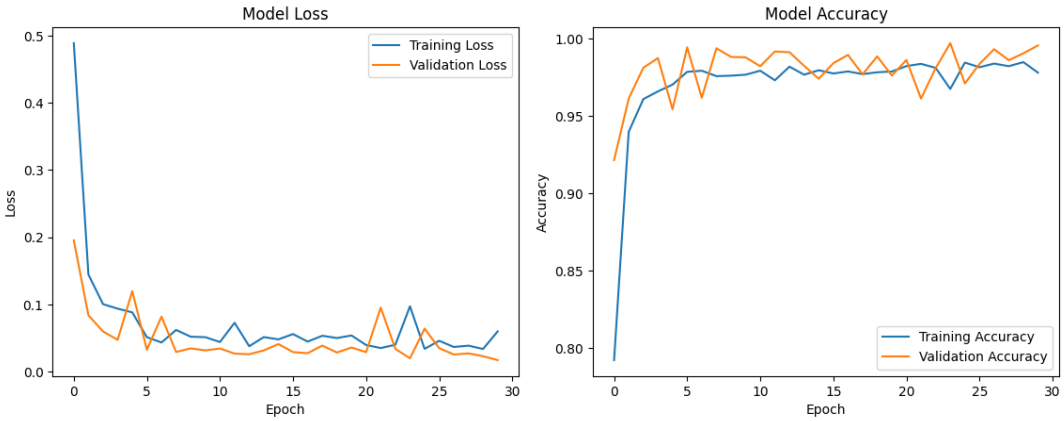
معیارهای عددی برای روش **8** جدول **KNN** برای **k=9** برای ورودی داده **scale**شده

AUC	0.79
Recall	0.79
F1	0.79
Precision	0.79

به وضوح طبقه بندی کننده بهبود یافته است.

MLP

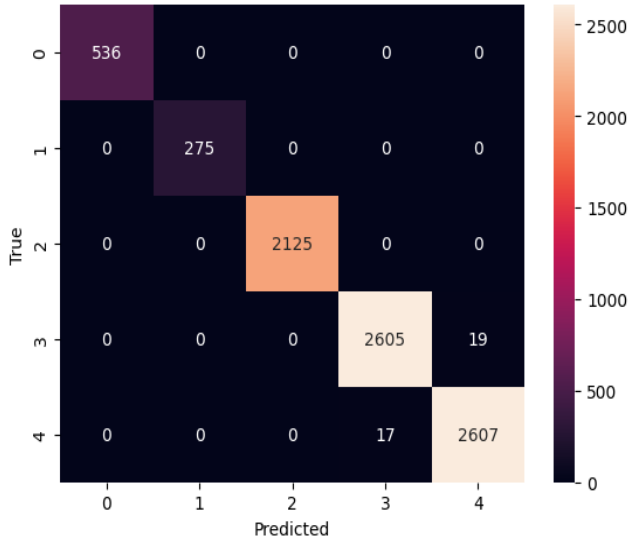
پس از اعمال scaler به داده‌ها و استفاده از شبکه قبلی، نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر بدست می‌آیند.



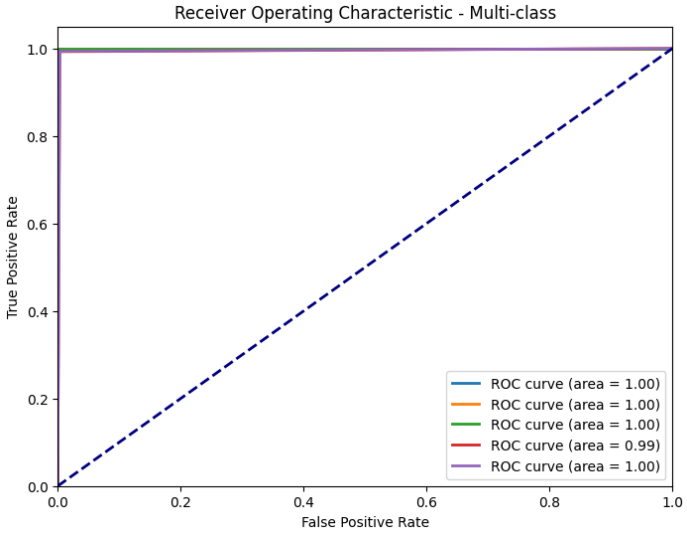
شکل ۲۱-نمودارهای هزینه و دقت برای شبکه **MLP** با ورودی داده **scale** شده

ماتریس درهم ریختگی به صورت زیر بدست می‌آید.

جدول ۹- ماتریس درهم ریختگی برای شبکه **MLP** با ورودی داده **scale** شده



نمودار ROC برای این روش به صورت زیر بدست آمده‌است.



شکل ۲۲- نمودار **ROC** برای شبکه **MLP** با ورودی داده **scale** شده

و در نهایت، مقادیر عددی زیر نیز تولید شدند.

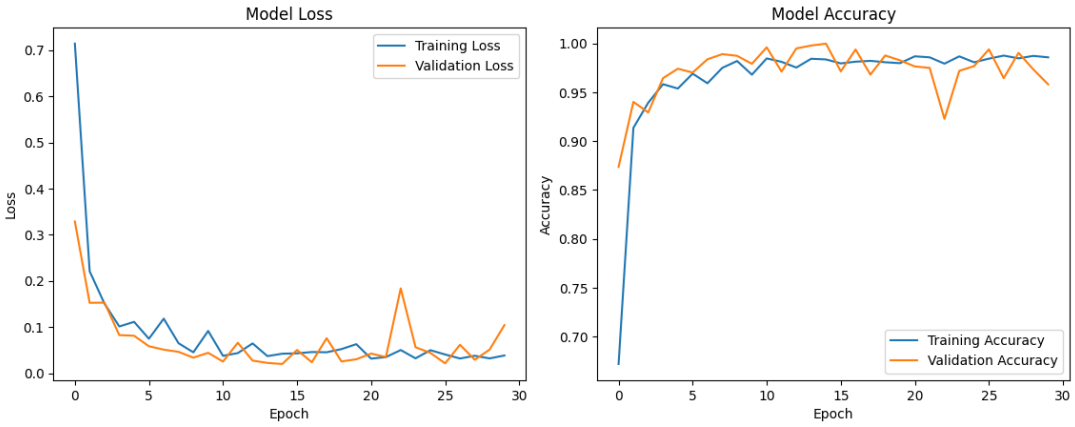
جدول ۱۰- معیارهای عددی برای شبکه **MLP** با ورودی داده **scale** شده

AUC	0.9979
Recall	0.9956
F1	0.9956
Precision	0.9956

و این امر تنها با استفاده از یک scaler انجام شد. مشخصا نشان می‌دهد که داده با کمک این scaler ویژگی‌های خود را بهتر نشان می‌دهد؛ به طوریکه برای شبکه قابل فهم‌تر شده تا جایی که توانسته با دقت خوبی آن‌ها را جدا کند.

CNN

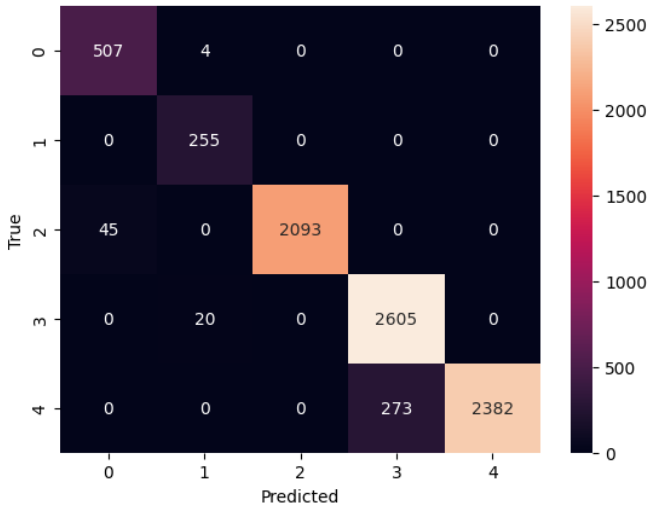
پس از اعمال scaler به داده‌ها و استفاده از شبکه قبلی، نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر بدست می‌آیند.



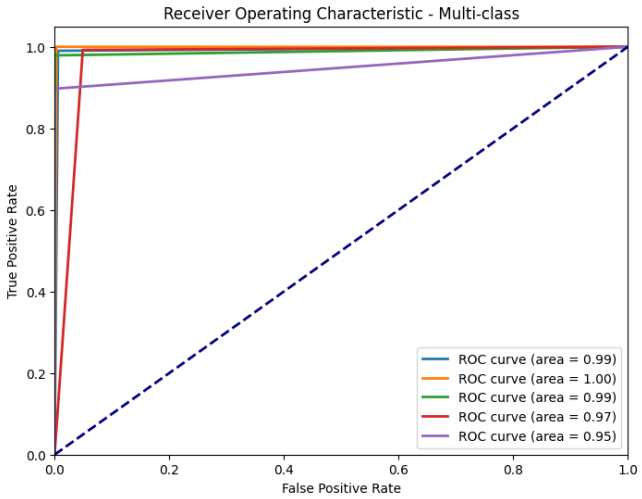
شکل ۲۳- نمودار هزینه و دقت برای شبکه **CNN** با ورودی داده **scale** شده

ماتریس درهم ریختگی به صورت زیر بدست می‌آید.

جدول ۱۰- ماتریس درهم ریختگی برای شبکه **CNN** با ورودی داده **scale** شده



نمودار ROC برای این روش به صورت زیر بدست آمده‌است.



شکل ۲۴- نمودار **ROC** برای شبکه **CNN** با ورودی داده **scale** شده

و مقادیر عددی زیر نیز تولید شدند.

جدول ۱۱- معیارهای عددی برای شبکه **CNN** با ورودی داده **scale** شده

AUC	0.98
Recall	0.95
F1	0.95
Precision	0.95

در این‌جا نیز، شبکه کاملا بهبود یافته و توانسته دقت خوبی از خود نشان دهد. هر چند که کمی ضعیف‌تر از MLP عمل کرده‌است.

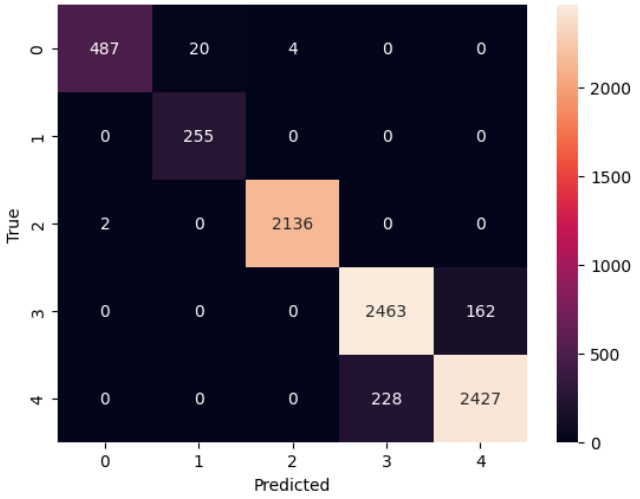
Hybrid

پس از اعمال scaler به داده‌ها و استفاده از شبکه قبلی، نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر بدست می‌آیند.

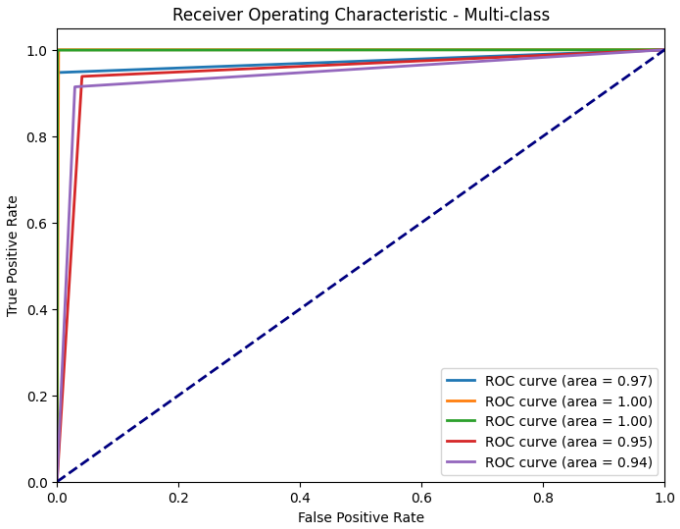
شکل ۲۵- نمودار هزینه و دقت برای شبکه **Hybrid** با ورودی داده **scale** شده

ماتریس درهم ریختگی به صورت زیر بدست می‌آید.

جدول ۱۱- ماتریس درهم ریختگی برای شبکه **Hybrid** با ورودی داده **scale** شده



نمودار ROC برای این روش به صورت زیر بدست آمده‌است.



شکل ۲۶- نمودار **ROC** برای شبکه **Hybrid** با ورودی داده **scale** شده

و مقادیر عددی زیر نیز تولید شدند.

جدول ۱۲- معیارهای عددی برای شبکه **Hybrid** با ورودی داده **scale** شده

AUC	0.97
Recall	0.94
F1	0.94
Precision	0.94

بازهم شبکه بهبود بسیار خوبی داشته، اما کمی از دو شبکه قبلی ضعیف‌تر کار کرده‌است.

مشکلی که در همه این شبکه‌های iterative استفاده شده می‌توان مشاهده کرد این است که شبکه در حین آموزش، دچار نوسانات بسیاری می‌باشد. این امر می‌تواند بر اثر انتخاب نادرست نرخ آموزش رخ دهد. از طرفی اگر یک نرخ آموزش یکسان برای کل فرایند آموزش در نظر بگیریم، احتمال وجود نوسانات در epoch های پایانی بیشتر می‌شود و دقت کلی شبکه پایین می‌آید و یا حتی شبکه overfit می‌شود. برای جلوگیری از این کار می‌توان از یک ماژول در شبکه به نام ReduceLROnPlateau استفاده کرد. از طرفی در صورتی که شبکه رو به overfit باشد، با اعمال کردن regularizer ها می‌توان از این امر جلوگیری کرد.

ReduceLROnPlateau

ReduceLROnPlateau یک callback در Keras (TensorFlow) است که به صورت پویا نرخ یادگیری را در طول آموزش بر اساس عملکرد مدل در مجموعه اعتبار سنجی تنظیم می کند. این به بهبود کارایی آموزش و دستیابی به همگرایی بهتر برای مدل های پیچیده کمک می کند.

ویژگی های اصلی آن، پاسخ به تماس به طور مداوم یک معیار اعتبار سنجی مشخص (به عنوان مثال، از میزان هزینه داده‌های اعتبار[1] یا دقت[2]) را برای تعیین اینکه آیا عملکرد مدل در حال بهبود است، نظارت می کند. به علاوه، امکان تنظیم مقدار "صبر کردن"[3] برا اساس معیار تحت نظارت وجود دارد؛ بدیت صورت که تعداد epoch هایی است که هیچ بهبودی ندارند، معلوم می‌شود و پس از آن ضریبی (معمولا کمتر از ۱) در نرخ آموزش ضرب شده و سرعت آموزش کاهش می یابد. یک پارامتر آستانه نیز تعریف می‌شود تا مقدار نرخ آموزش از آن کمتر نشود. ReduceLROnPlateau به عنوان آرگومان به پارامتر callbacks در تابع model.fit ارسال شده و در طول کامپایل مدل استفاده می‌شود.

این روش به تعدیل نرخ یادگیری پویا به مدل کمک می کند تا به طور کارآمدتر همگرا شود و به نتایج آموزشی بهتری دست یابد. اما از طرفی، با کاهش نرخ یادگیری ممکن است مانع از دستیابی مدل به راه حل بهینه شود.

کد این روش به صورت زیر می‌باشد.

```
from keras.callbacks import  ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.8,
                              patience=3,
                              min_lr=0.0001, verbose=1)

history = model.fit(x_train, y_train, batch_size=32, epochs=40, validation_data=(x_test, y_test), callbacks=[reduce_lr])
```

شکل ۲۷- کد مربوط به **ReduceLROnPlateau**

[1] Validation loss
[2] Validation accuracy
[3] Patience

Regularizer

تنظیم‌کننده هسته و تنظیم‌کننده بایاس، تکنیک‌هایی هستند که در یادگیری ماشین برای جلوگیری از overfitting و بهبود عملکرد تعمیم یک مدل[1] استفاده می‌شوند. منظم‌سازی هنگام برخورد با مدل‌های پیچیده ضروری است تا از تخصصی شدن بیش از حد مدل به داده‌های آموزشی جلوگیری شود، بنابراین توانایی آن برای تعمیم به داده‌های جدید و نادیده را قربانی می‌کند.

تنظیم کننده هسته که به عنوان تنظیم کننده وزن یا کاهش وزن نیز شناخته می شود، در طول تمرین بر روی هسته (وزن) مدل اعمال می شود. یک عبارت جریمه به تابع ضرر اضافه می کند که متناسب با بزرگی وزن ها است. با انجام این کار، منظم کننده مدل را تشویق به یادگیری وزنه های کوچکتر می کند، به طور موثری پیچیدگی مدل را کاهش می دهد و از تطبیق آن با نویز در داده های آموزشی جلوگیری می کند. منظم سازی L1 و تنظیم L2 دو نوع متداول تنظیم کننده هسته هستند. منظم‌سازی L1 جریمه‌ای متناسب با مقادیر مطلق وزن‌ها را معرفی می‌کند، که باعث ایجاد پراکندگی در مدل می‌شود. از سوی دیگر، منظم‌سازی L2 مقادیر مجذور وزن‌ها را جریمه می‌کند و وزن‌های کوچک‌تر اما غیر صفر را ارتقا می‌دهد. معادله‌های آن‌ها به صورت زیر می‌باشد.

$$L1: loss = e + \lambda \sum |w_i|$$
$$L2: loss = e + \lambda \sum w_i^2$$

تنظیم‌کننده بایاس شبیه تنظیم‌کننده هسته است، اما به‌جای وزن‌ها، برای اصطلاحات بایاس مدل اعمال می‌شود. عبارات بایاس به خروجی هر نورون در یک شبکه عصبی اضافه می‌شوند و به مدل اجازه می‌دهند تا افست یا تغییر در داده‌ها را ثبت کند. به کارگیری منظم‌سازی سوگیری به کاهش تعصب نسبت به کلاس‌ها یا الگوهای خاص کمک می‌کند و منجر به پیش‌بینی‌های متعادل‌تر و دقیق‌تر می‌شود.

$$L1: new\ bias = old\ bias + \lambda \sum |b_i|$$
$$L2: new\ bias = old\ bias + \lambda \sum b_i^2$$

با استفاده از تنظیم‌کننده‌های هسته و بایاس، مدل قوی‌تر می‌شود، بهتر به داده‌های دیده نشده تعمیم می‌یابد، و احتمال اینکه در مجموعه آموزشی اضافه شود، کمتر است. این تنظیم‌کننده‌ها تعادلی بین برازش داده‌های آموزشی و کنترل پیچیدگی مدل ایجاد می‌کنند و عملکرد بهتری را در داده‌های جدید و دیده نشده تضمین می‌کنند.

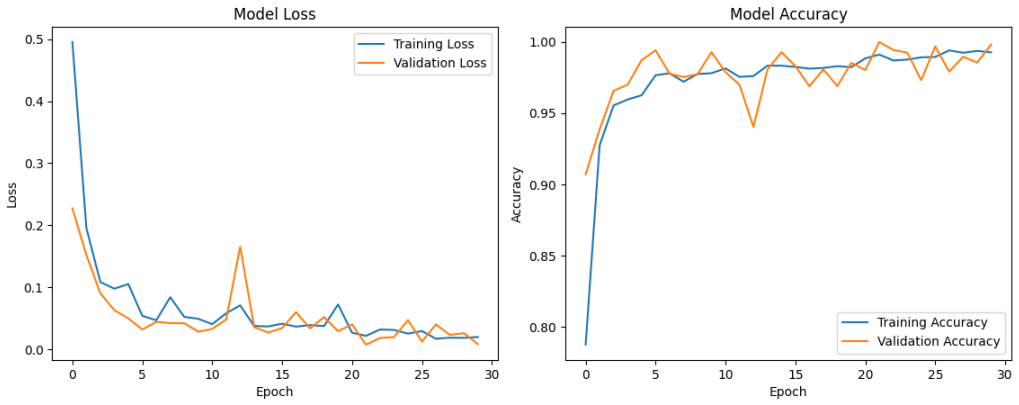
[1] Generalizing the model

بررسی روش‌ها

ماژول ReduceLROnPlateau روی همه شبکه‌ها اعمال شد اما regularizer ها فقط در ۲ قسمت اعمال شدند که در جای آن، توضیح داده خواهد شد.

MLP

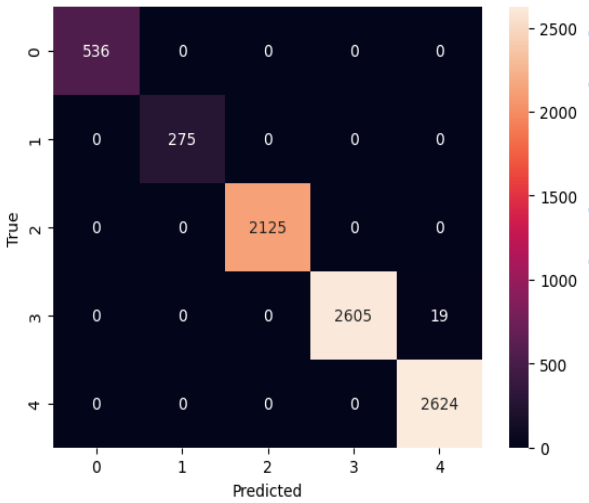
پس از اعمال ReduceLROnPlateau روی شبکه با ضریب اعمال ۰.۹ برای ۲ epoch صبر کردن، نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر خواهد بود.



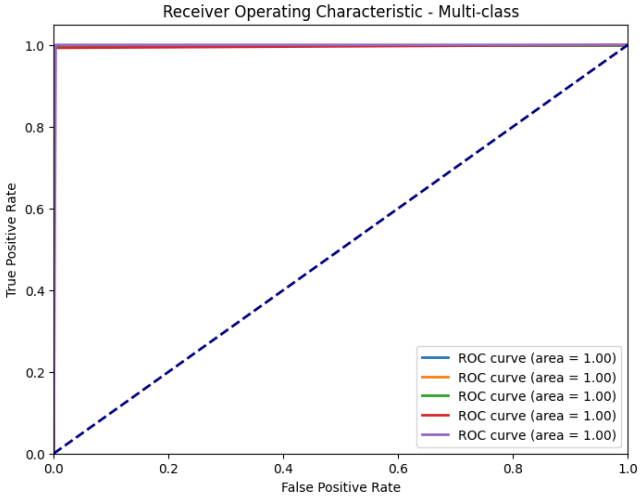
شکل ۲۸- نمودار هزینه و دقت برای شبکه **MLP** با **ReduceLROnPlateau**

مشاهده می‌شود که نمودار هزینه، به وضوح بهبود یافته. هرچند که هنوز کمی نوسانات وجود دارد. این موضوع می‌تواند با انتخاب درست نرخ آموزش اولیه پاسخ بهتری دریافت کرد. ماتریس درهم ریختگی آن به صورت زیر می‌باشد.

جدول ۱۲-ماتریس درهم ریختگی برای شبکه **MLP** با **ReduceLROnPlateau**



نمودار ROC آن به صورت زیر می‌باشد.



شکل ۲۹- نمودار **ROC** برای شبکه **MLP** با **ReduceLROnPlateau**

سطح زیر نمودار هر کلاس، تقریبا برابر ۱ شده که این بهترین حالت است. مقادیر زیر نیز بدست آمده‌اند.

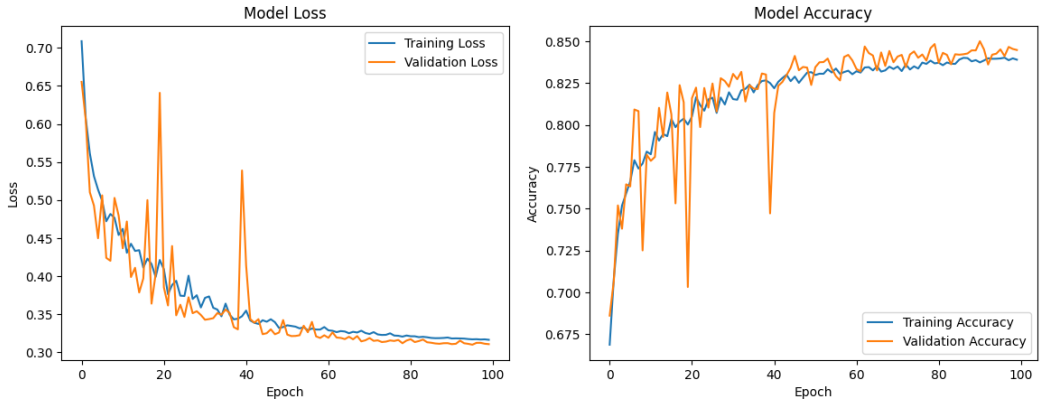
جدول ۱۳- معیارهای عددی برای برای شبکه **MLP** با **ReduceLROnPlateau**

AUC	0.9989
Recall	0.9976
F1	0.9976
Precision	0.9976

این مقادیر نیز اندکی بهبود داشته‌اند.

MLP on LDA output

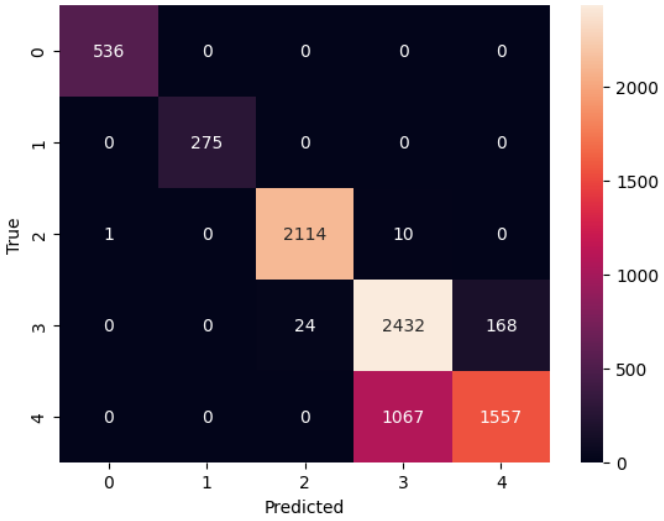
به دلیل اینکه اعمال scaler روی داده‌های این روش کافی نبوده، هم ReduceLROnPlateau با ضریب اعمال ۰.۸ برای ۳ epoch صبر کردن و هم regularizer در لایه مخفی دوم با مقدار λ برابر ۰.۰۰۱ هم برای کرنل هم برای بایاس درنظر گرفته شده‌است. علت انتخاب این مقدار این است که معمولا مدل‌هایی که با این مقدار ایجاد می‌شوند، بهترین آن هستند. نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر می‌باشد.



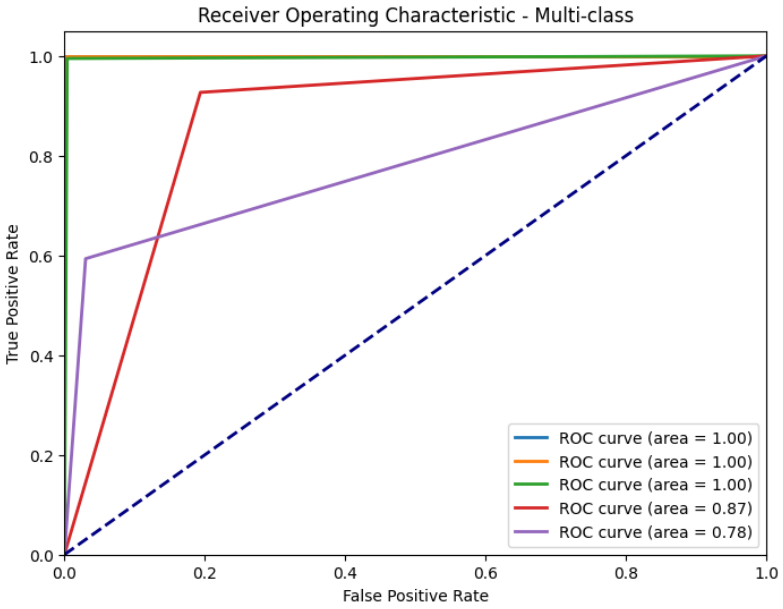
شکل ۳۰-نمودار هزینه و دقت برای شبکه **MLP** با **ReduceLROnPlateau** روی خروجی **LDA**

ماتریس درهم ریختگی به صورت زیر بدست آمده‌است.

جدول ۱۳- ماتریس درهم ریختگی برای شبکه **MLP** با **ReduceLROnPlateau** روی خروجی **LDA**



و نمودار ROC نیز به صورت زیر بدست می‌آید.



شکل ۳۱- نمودار **ROC** برای شبکه **MLP** با **ReduceLROnPlateau** روی خروجی **LDA**

و در نهایت مقادیر زیر نیز تولید شده‌اند.

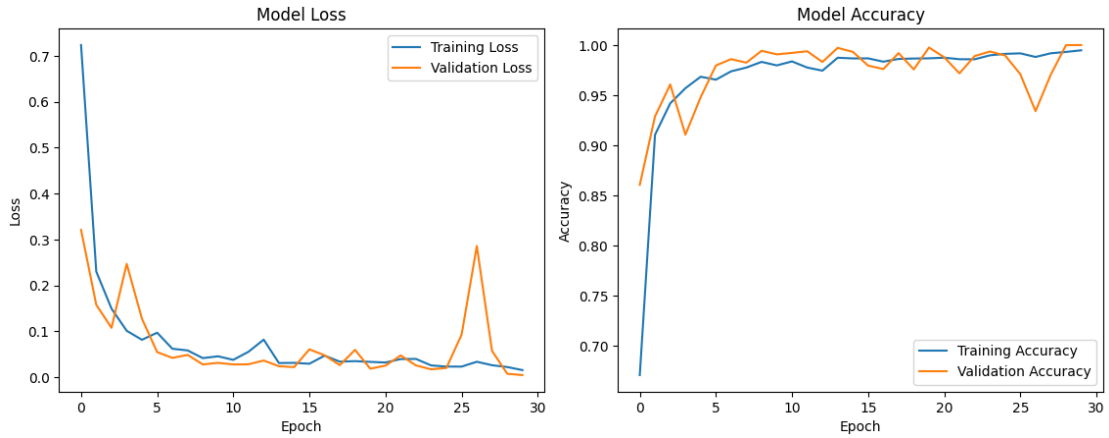
جدول ۱۴- معیارهای عددی برای شبکه **MLP** با **ReduceLROnPlateau** روی خروجی **LDA**

AUC	0.92
Recall	0.84
F1	0.84
Precision	0.84

شبکه به حد اشباع رسیده و نمی‌توانست بهتر از این نتیجه‌ای را ارائه دهد. این امر نشان می‌دهد که با وجود اینکه کاهش بعد روی داده‌ها اعمال شده، ویژگی‌های مهم شبکه، خصوصا ویژگی‌هایی که کلاس ۴ و ۵ را از هم جدا می‌کند، از بین رفته‌اند.

CNN

پس از اعمال ReduceLROnPlateau روی شبکه با ضریب اعمال ۰.۹ برای ۳ epoch صبر کردن، نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر خواهد بود.



شکل ۳۲- نمودارهای هزینه و دقت برای شبکه **CNN** با **ReduceLROnPlateau**

به صورت کلی نمودارها کمی روان‌تر شده است؛ اما یک جهش ناگهانی در epoch های پایانی وجود دارد. verbose حین اجرا برای این قسمت به صورت زیر می‌باشد.

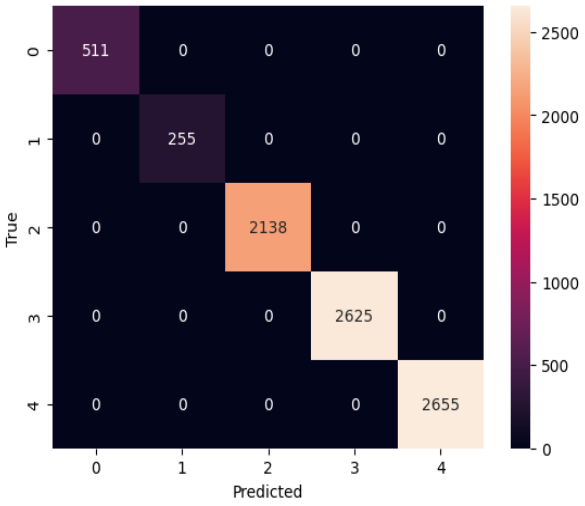
```
Epoch 24/30
597/597 [==============================] - 3s 4ms/step - loss: 0.0255 - accuracy: 0.9897 - val_loss: 0.0171 - val_accuracy: 0.9935 - lr: 0.0044
Epoch 25/30
597/597 [==============================] - 3s 5ms/step - loss: 0.0231 - accuracy: 0.9912 - val_loss: 0.0200 - val_accuracy: 0.9896 - lr: 0.0044
Epoch 26/30
597/597 [==============================] - 2s 4ms/step - loss: 0.0229 - accuracy: 0.9917 - val_loss: 0.0924 - val_accuracy: 0.9709 - lr: 0.0044
Epoch 27/30
586/597 [============================>.] - ETA: 0s - loss: 0.0337 - accuracy: 0.9881
Epoch 27: ReduceLROnPlateau reducing learning rate to 0.0039366002194583411.
597/597 [==============================] - 2s 4ms/step - loss: 0.0335 - accuracy: 0.9881 - val_loss: 0.2857 - val_accuracy: 0.9340 - lr: 0.0044
Epoch 28/30
597/597 [==============================] - 3s 4ms/step - loss: 0.0260 - accuracy: 0.9918 - val_loss: 0.0574 - val_accuracy: 0.9704 - lr: 0.0039
Epoch 29/30
597/597 [==============================] - 3s 6ms/step - loss: 0.0220 - accuracy: 0.9931 - val_loss: 0.0074 - val_accuracy: 0.9999 - lr: 0.0039
Epoch 30/30
597/597 [==============================] - 3s 4ms/step - loss: 0.0154 - accuracy: 0.9948 - val_loss: 0.0045 - val_accuracy: 1.0000 - lr: 0.0039
Training Time: 82.53 seconds
```

شکل ۳۳- بخشی از **verbose** کد در هنگام آموزش

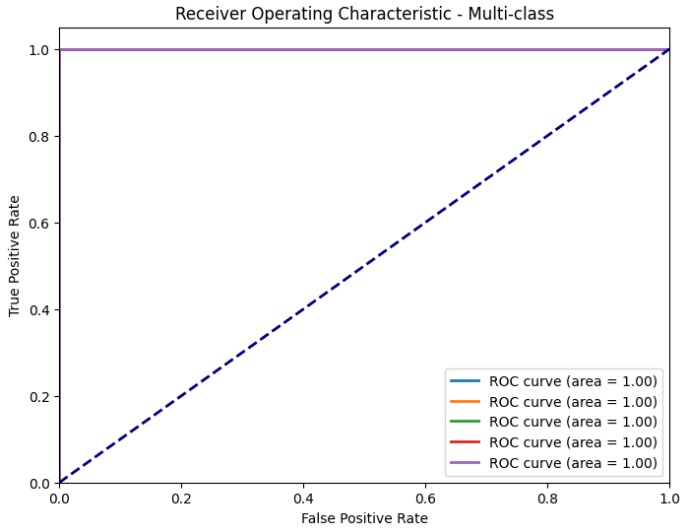
در epoch ۲۷ ام، بعد از اینکه شبکه شروع به overfit کردن می‌کند، با اعمال ضریب به نرخ آموزش، از رخ دادن این اتفاق جلوگیری کرده و پس از آن شبکه در حال بهبود بوده است. در اصل با نرخ آموزش ۰.۰۰۴۴ شبکه در epoch ۲۷ ام در حال overfit شدن بوده که این ماژول از این اتفاق جلوگیری کرده‌است. هرچند که تعداد epoch ها در همه اجراها ۳۰ بوده، اما اگر این تعداد را افزایش دهیم، اتفاقی مشابه در epoch های بعدی رخ می‌دهد. در اصل شبکه در نزدیک‌ترین حالت به جواب است و دیگر نیازی به ادامه نیست؛ در غیر این صورت نیاز است تا نرخ آموزش بعد از این epoch در حین آموزش به صفر همگرا شود.

ماتریس درهم ریختگی شبکه به صورت زیر می‌باشد.

جدول ۱۴- ماتریس درهم ریختگی برای شبکه **CNN** با **ReduceLROnPlateau**



نمودار ROC به صورت زیر می‌باشد.



شکل ۳۴- نمودار**ROC** برای شبکه **CNN** با **ReduceLROnPlateau**

و مقادیر زیر نیز بدست آمده‌اند.

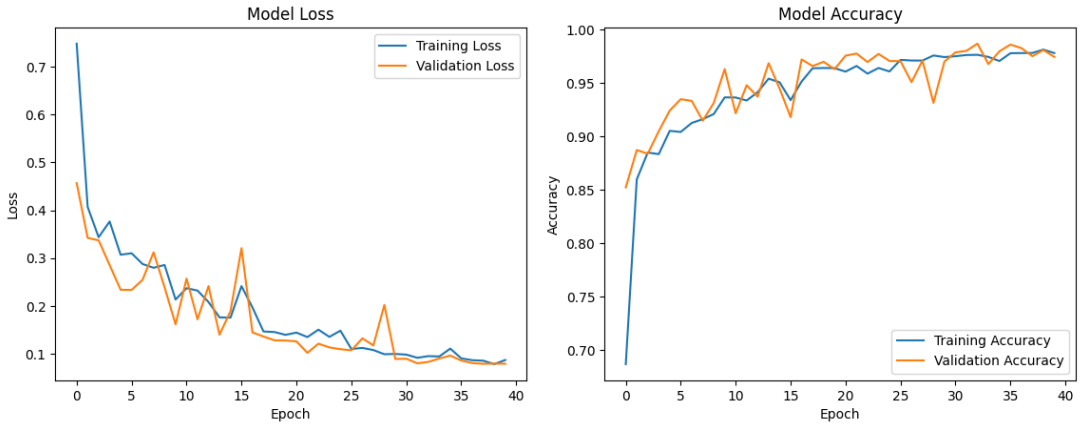
جدول ۱۵- معیارهای عددی برای شبکه **CNN** با **ReduceLROnPlateau**

AUC	1.0
Recall	1.0
F1	1.0
Precision	1.0

شبکه علاوه بر اینکه توانسته خطای سیستم را به خوبی از حالت سالم جدا کند، هر کدام از خطاها نیز کاملا به درستی تشخیص داده شده‌اند.

Hybrid

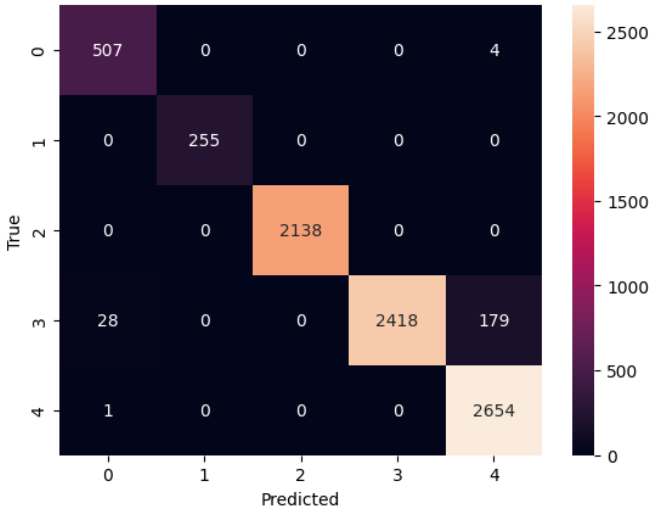
به دلیل اینکه اعمال scaler روی داده‌های این روش کافی نبوده، هم ReduceLROnPlateau با ضریب اعمال ۰.۸ برای ۳ epoch صبر کردن و هم regularizer در لایه LSTM با مقدار λ برابر ۰.۰۰۱ هم برای کرنل هم برای بایاس درنظر گرفته شده‌است. نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر می‌باشد.



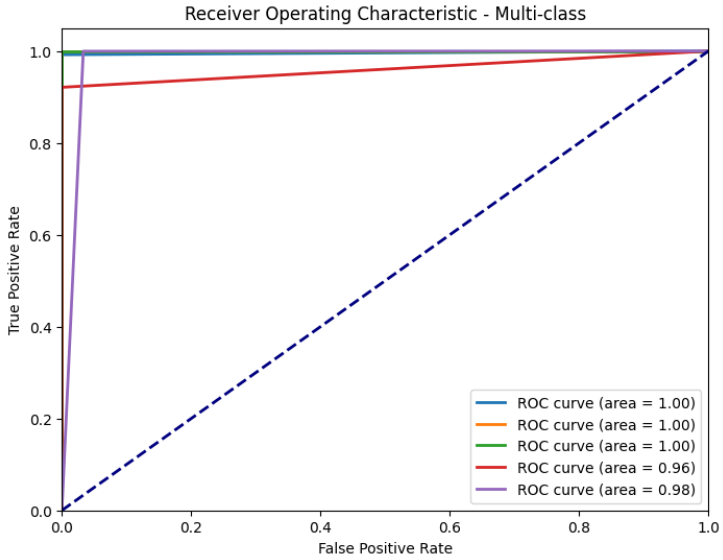
شکل ۳۵- نمودار هزینه و دقت برای شبکه **Hybrid** با **ReduceLROnPlateau** و **regualizer**

به وضوح نوسانات کاهش یافته‌است. ماتریس درهم ریختگی شبکه به صورت زیر می‌باشد.

جدول ۱۵- ماتریس درهم ریختگی برای شبکه **Hybrid** با **ReduceLROnPlateau** و **regualizer**



نمودار ROC به صورت زیر می‌باشد.



شکل ۳۶- نمودار **ROC** برای شبکه **Hybrid** با **ReduceLROnPlateau** و **regualizer**

و مقادیر زیر تولید شدند.

جدول**16** - معیارهای عددی برای شبکه **Hybrid** با **ReduceLROnPlateau** و **regualizer**

AUC	0.98
Recall	0.97
F1	0.97
Precision	0.97

شبکه بهبود خوبی از خود نشان داده؛ هرچند باز هم از دو شبکه دیگر ضعیف‌تر عمل کرده‌است.

شبکه Transformer

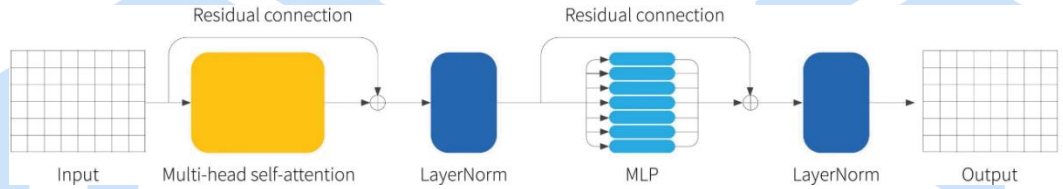
ترنسفورمرها به شدت حوزه‌ی پردازش زبان طبیعی را تحت تأثیر قرار داده و نقش قابل توجهی در وظایف مختلف طبقه‌بندی ایفا کرده‌اند. این مدل‌ها در ابتدا در مقاله "Attention is All You Need" از واسوانی و همکاران در سال ۲۰۱۷ معرفی شدند و بر اساس مکانیسم خودتوجه[1] عمل می‌کنند.

در وظایف طبقه‌بندی، ترنسفورمرها به دلیل قابلیت آنها برای درک وابستگی‌ها و روابط بین کلمات یا توکن‌ها در یک دنباله، بسیار موثر ثابت شده‌اند. در مقابل شبکه‌های عصبی دنباله‌ای بازگشتی (RNN) و شبکه‌های عصبی کانولوشنال (CNN)، ترنسفورمرها دنباله ورودی را به طور همزمان پردازش می‌کنند که آنها را برای متون بلند موثر و کارآمد می‌سازد.

[1] Self-attention

اجزاء کلیدی ترنسفورمرها، مکانیزم توجه[1] است که به مدل امکان تمرکز بر بخش‌های مرتبط و مهم ورودی هنگام تولید خروجی می‌دهد. توجه به خود، ترنسفورمر را قادر می‌سازد تا به هر توکن وزن‌های متفاوتی اختصاص دهد که بر اساس اهمیت آن در وظیفه مورد نظر است و روابط بین توکن‌های دوردست را به طور موثری درک می‌کند. این رویکرد مبتنی بر توجه به ترنسفورمر کمک می‌کند تا بهترین فهم از متن ورودی وزن بگیرد و عملکرد طبقه‌بندی را بهبود ببخشد.

یکی از معماری‌های معروف ترنسفورمرها برای وظایف پردازش زبان طبیعی مدل "BERT" (پردازش همزمان دوجهته با نمایش‌ها از ترنسفورمرها) است. BERT با تربیت پیش‌فرض بر روی مجموعه‌داده‌های بزرگ و سپس با اجرای مراحل زیروظایف برای وظایف خاصی مانند تحلیل احساسات، طبقه‌بندی متن و تشخیص نهادهای نام‌دار، مورد استفاده قرار می‌گیرد. طبیعت دوجهته آن به آن امکان می‌دهد تا متن را از طرفین یک توکن دریافت کرده و توانایی درک معنای متن را از دو جهت چپ و راست توکن تقویت کند و این موضوع باعث بهبود کارآیی در درک مفهوم متن می‌شود.



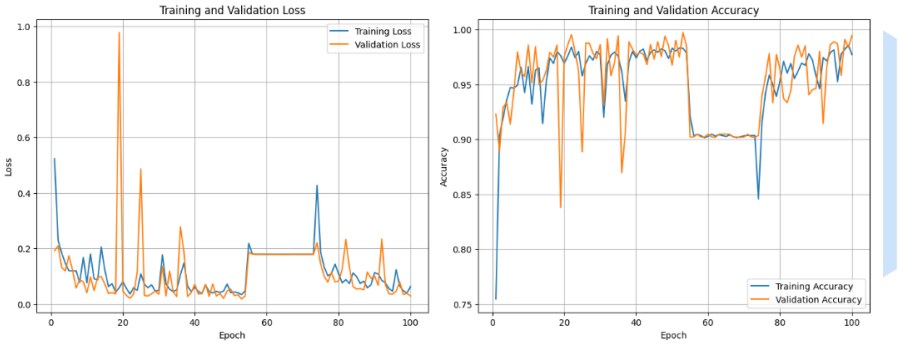
شکل ۳۷ - نمایی از روش **Transformer**

در این پروژه سعی شد تا نمونه بسیار ساده‌ای از آن بر روی داده‌های scale شده بررسی شود. ابتدا یک تابع transformer_encoder تعریف می‌شود که معماری Transformer Encoder را می‌سازد. این تابع دو آرگومان می‌گیرد: input_shape که شکل داده‌های ورودی را مشخص می‌کند، و num_classes که تعداد کلاس‌ها برای کار طبقه بندی را مشخص می‌کند. در داخل تابع، یک لایه با ۳۲ نرون و تابع فعال‌سازی relu پس از لایه ورودی اضافه می‌شود.پس از آن، یک لایه دیگر با ۱۶ نرون و تابع فعال‌سازی relu پس از لایه قبلی اضافه می‌شود. بعد از آن، کد داده‌ها را تغییر شکل می‌دهد و یک بعد جدید، صرفا برای اجرای مرحله بعد، به آن اختصاص می‌دهد و مکانیزم ساده شده از خود-توجه (Scaled Dot-Product Attention) را اجرا می‌کند. اما حائز اهمیت است که این پیاده‌سازی اجزای کلیدی مانند کدگذاری مکانی[2] و خود-توجه چندلایه‌ای[3] ترانسفورمر را ندارد. خود-توجه چندلایه‌ای را بر روی داده‌های ورودی x با دو سر خود-توجه (num_heads=2) اعمال می‌کند. پس از آن بعد اضافه شده به داده‌ها در قبل را حذف می‌شود و به

[1] Attention
[2] positional encoding
[3] multi-layer self-attention

حالت ۲ بعدی بازشکل می‌دهد. بعد از اعمال مکانیزم ساده خود-توجه، لایه‌های چگال بیشتری اضافه می‌شود تا ویژگی‌های استخراج شده از گام خود-توجه را بهبود بخشد. این لایه یک لایه با ۸ نرون و تابع فعال‌سازی relu پس از خروجی خود-توجه اضافه می‌شود. لایه چگال نهایی با num_classes نرون و تابع فعال‌سازی softmax احتمال‌های خروجی برای هر کلاس را تولید می‌کند. این لایه خروجی مدل است.

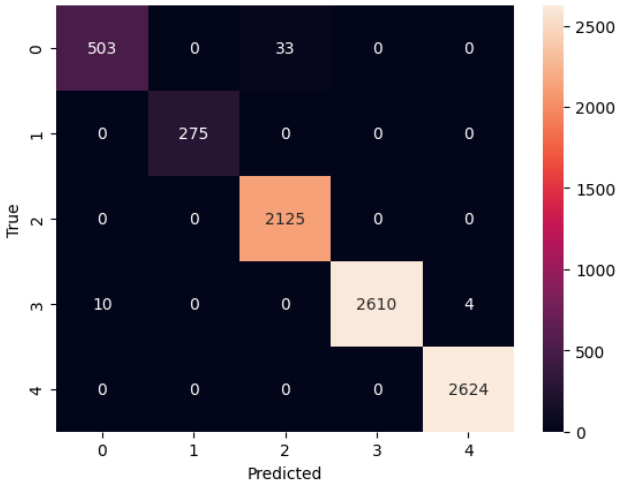
تابع transformer_encoder مدل ساخته‌شده را برمی‌گرداند. سپس، کد مدل را با بهینه‌سازی‌کننده ADAM با نرخ یادگیری ۰.۰۱ آموزش دیده و از تابع هزینه Sparse Categorical Crossentropy برای کار طبقه بندی استفاده می‌شود. معیار دقت Categorical Sparse Accuracy نیز برای نظارت بر عملکرد مدل در طول آموزش استفاده می‌شود. سپس وارد حلقه‌ی for می‌شود و مدل را برای حداکثر ۱۰۰ دوره، به جای ۱۰۰ epoch در یک آموزش، آموزش می‌دهد. هزینه و دقت آموزش و همچنین هزینه و دقت اعتبارسنجی برای هر اپوک ثبت می‌شوند تا بعد از اتمام کار رسم گردند. با استفاده از مکانیزم قطع زودهنگام تعریف‌شده توسط تابع EarlyStopping، حلقه به طور زودهنگام متوقف می‌شود؛ اگر هیچ بهبودی در هزینه اعتبارسنجی برای ۱۰ epoch متوالی اتفاق نیفتد. در نهایت نمودارهای هزینه و دقت داده‌های آموزش و اعتبارسنجی به صورت زیر بدست می‌آیند.



شکل ۳۸- نمودار هزینه و دقت برای شبکه **Transformer**

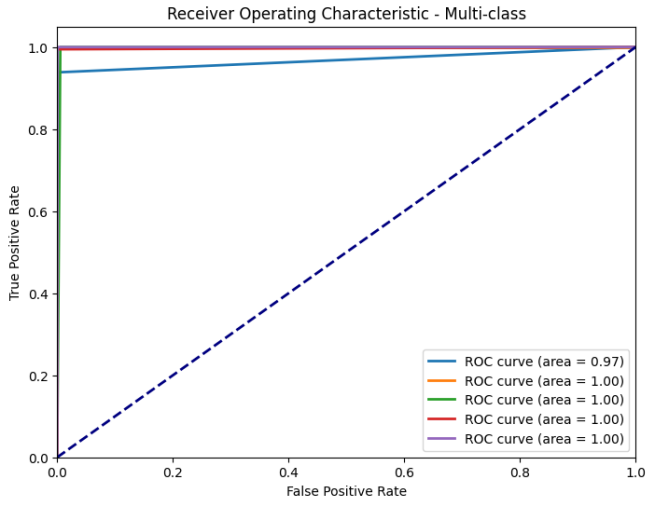
مشاهده می‌شود شبکه متفاوت‌تر از سایر شبکه‌ها اجرا شده است. همچنین یک تغییر ناگهانی در شبکه دیده می‌شود که البته کاملا در ادامه جبران شده و کار شبکه خاتمه یافته‌است. ماتریس درهم ریختگی به صورت زیر بدست آمده‌است.

جدول ۱۷- ماتریس درهم ریختگی برای شبکه **Transformer**



نکته جالبی که وجود دارد این است که برخلاف سایر شبکه‌ها، اشتباهات تشخیصی در کلاس اول بیشتر است. این در حالی است که شبکه‌های قبلی در کلاس‌های ۴ و ۵ بیشترین اشتباه را از خود نشان می‌دادند. بنابراین این احتمال وجود دارد که اگر داده‌های بیشتری از دو کلاس اول و خصوصا کلاس اول در دسترس باشد، اشتباهات شبکه در این کلاس نیز بشدت کاهش پیدا کند.

نمودار ROC نیز به صورت زیر می‌باشد.



شکل ۳۹- نمودار **ROC** برای شبکه **Transformer**

در این نمودار هم ضعفی که کلاس اول در ماتریس درهم ریختگی نشان داد، به نحو دیگری قابل مشاهده می‌باشد.

در نهایت، مقادیر زیر نیز بدست آمده‌اند.

جدول**18** - معیارهای عددی برای شبکه **Transformer**

AUC	0.99
Recall	0.99
F1	0.99
Precision	0.99

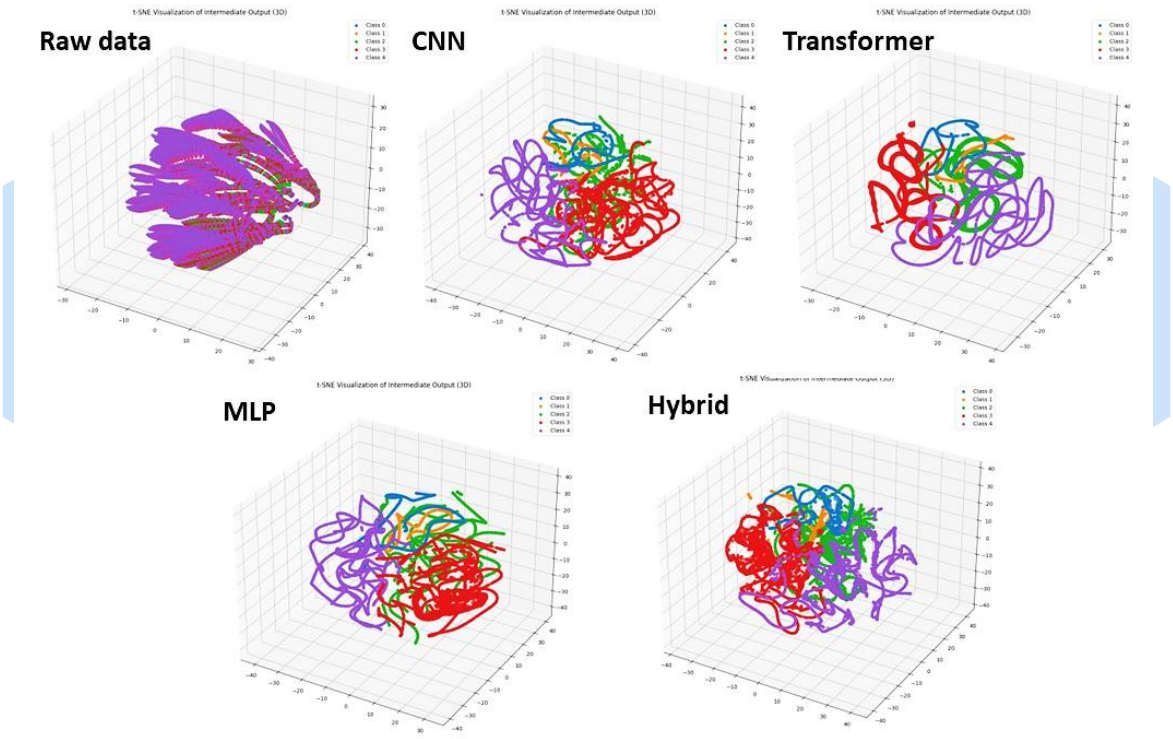
شبکه، عملکرد بسیار مناسبی را از خود نشان داده‌است. دقت کنید که تمامی خروجی‌های بدست آمده تنها با یک scaler بوده و دو روش دیگر، اعمال نشده‌اند.

بخش ۳: جمع بندی نتایج و مقایسه

راه‌های بسیاری برای مقایسه خروجی‌های هر شبکه وجود دارد. یکی از آن‌ها استفاده از تجسم کننده t-SNE می‌باشد.

نمایش t-SNE

در ابتدا دیدیم که داده‌های خام به چه صورت دیده شدند. حال اگر بخواهیم خروجی‌های شبکه‌ها را نمایش دهیم، نیاز است که خروجی میانی از یک لایه پنهان (قبل از لایه خروجی نهایی) استخراج شده و سپس t-SNE را اعمال شود تا ابعاد آن به سه بعدی کاهش یابد. شکل زیر نمایی از تمامی خروجی‌های شبکه‌ها در کنار خروجی داده‌های خام می‌باشد.

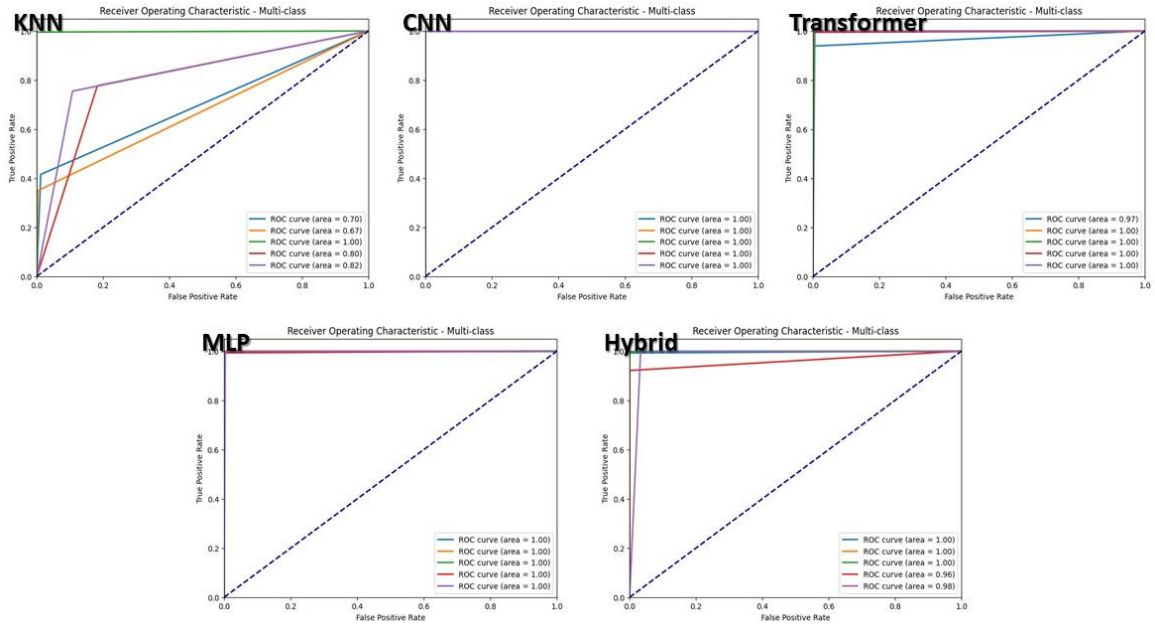


شکل ۴۰- مقایسه خروجی‌های مختلف هر شبکه در محیط **t-SNE** نسبت به داده خام

خروجی‌های بدست آمده، نشان می‌دهند که هر کدام چطور داده‌ها را از هم جدا کرده‌اند. نکته جالب آن این است که شبکه ترانسفورمر با وجود پیچیدگی بیشتر، ظاهر ساده‌تر و متمایزتری برای هر کلاس روی داده‌ها ایجاد کرده‌است.

نمودارهای ROC

از راه‌های ساده و متداول مقایسه، مقایسه نمودارهای ROC هر شبکه می‌باشد. این نمودارها به صورت زیر می‌باشند.



شکل ۴۱- مقایسه نمودارهای **ROC** بدست آمده در هر روش

در این نمودارها، شبکه‌ها CNN و MLP بهترین خروجی را دارند و پس از آن ترنسفورمر عملکرد بهتری دارد.

مقایسه عددی شبکه‌ها

در حین پردازش داده‌ها، معیارهای عملکردی مختلفی نیز مورد بررسی قرار گرفتند که در جدول زیر قابل مشاهده می‌باشند. اولین معیار، معیار زمان صرف شده برای اجرای هر شبکه است. معیار دوم آخرین مقدار هزینه‌ی داده‌های اعتبارسنجی در شبکه است. معیار سوم، تکرار پذیری هر شبکه است؛ بدیت صورت که در هر تکرار به صورت نسبی چقدر پاسخ ها به هم نزدیک‌تر بوده‌است که به صورت رتبه بندی نشان داده شده‌است. معیار چهارم، معیار زمان صرف شده برای هر روش یک معیار نسبی برای بررسی میزان زمان صرف شده برای توسعه هر شبکه برای رسیدن به بهترین پاسخ خود می‌باشد که به صورت رتبه بندی نشان داده شده‌است. معیار آخر نیز میزان نسبی سادگی شبکه برای رسیدن به بهترین پاسخ خود می‌باشد.

جدول ۱۹- مقایسه معیارهای عددی و نسبی هر شبکه

	Duration(s)	Validation Loss	Repeatability	Time spent on development	Network simplicity
MLP	82.37	0.0065	4	2	1
CNN	82.53	0.0045	2	4	2
Hybrid	143.52	0.0788	3	1	3
Transformer	400.32	0.0286	1	3	4

با مقایسه مقادیر و نمودارهای بالا می‌توان متوجه شد که در بین شبکه‌های فعلی بهترین شبکه، CNN می‌باشد. این شبکه با وجود اینکه کمی دیرتر از MLP اجرا می‌شود، اما اختلاف آن‌ها قابل چشم پوشی است. از طرفی، کمترین مقدار را در ستون دوم دارد و عملکرد خوبی در تکرارپذیری از خود نشان داده‌است. زمان صرف شده برای رسیدن به شبکه مناسب نیز برای آن زیاد نبوده و به نسبت شبکه ساده‌است و نیاز به سیستم‌های پیچیده و سنگین برای پردازش ندارد.

پیشنهادات

در بخش ترانسفورمر به این موضوع اشاره شد که داده‌های کلاس اول و دوم تعداد کمتری نسبت به سایرین دارند. هرچند که داده‌های کلاس دوم به نسبت متمایز تر از سایر داده‌هاست، اما تعداد کم آن می‌تواند مشکل ساز باشد. یکی از روش‌هایی که می‌توان این اختلاف بین تعداد داده‌ها را جبران کرد، استفاده از شبکه‌هایی مثل شبکه‌های مولد متخاصم[1] و یا خودرمزگذار متغیر[2] می‌باشد. این شبکه‌ها به نسبت روش یک جمع آوری داده معمولا زمان بسیار کمتری نیاز دارند تا داده‌های جدید را تولید کنند.

[1] Generative Adversarial Network (GAN)
[2] Variational Autoencoder (VAE)

هرچند که دارای فرآیندهای آموزشی پیچیده هستند و در تولید نمونه های با کیفیت بالا و متنوع دارای مشکلاتی هستند. ازطرفی به دلیل اینکه داده‌های جدید براساس داده‌های قبلی تولید می‌شوند، امکان سوگیری‌های در توزیع داده‌ها وجود دارد.

در کنار آن دیدیم که شبکه ترانسفورمر عملکرد متفاوت و دقیق‌تر را نسبت به سایرین داشت. اگر زمان بررسی و سیستم پردازشی، مورد حائز اهمیت نباشد، می‌توان با بهبود این شبکه به بهترین جواب رسید؛ چراکه این شبکه بهترین تکرارپذیری را دارد و در هر اجراف پاسخی مشابه را ارائه می‌دهد. این نشان دهنده قابل اعتماد بودن خروجی این شبکه می‌باشد.

مورد مهم دیگری که در ابتدای پروژه وجود داشت این بود که روش مناسبی برای کاهش بعد داده‌ها پیدا نشد. اگر بتوان روشی مناسب برای کاهش بعد پیدا کرد که ویژگی‌های مهم داده‌ها را نگه دارد، می‌توان شبکه‌های پیچیده‌تر و بهتری مثل ترنسفورمر را روی آن اجرا کرد تا هم دقت شبکه بالا باشد و هم سرعت رسیدن به پاسخ بالاتر برود.

مراجع

https://blog.keras.io/building-autoencoders-in-keras.html

https://ieee-dataport.org/open-access/simulated-boiler-data-fault-detection-and-classification

https://medium.com/@nagam808surya/understanding-overfitting-using-higher-order-linear-regression-31a0c7137aae

https://towardsdatascience.com/transformers-141e32e69591