به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین پنجم

پرسش ۱	نام و نام خانوادگی	آرمان مجیدی
	شماره دانشجویی	810100205
پرسش ۲	نام و نام خانوادگی	آرین فیروزی
	شماره دانشجویی	810100196
	مهلت ارسال پاسخ	**۱۴۰۱.10.13**

فهرست

پرسش ۱. پیش‌بینی نیروی باد به کمک مبدل و تابع خطای Huber 1

1-1. مقدمه 1

1-2. آماده‌سازی 1

1-3. روش‌شناسی و نتایج 3

پرسش 2 – استفاده از ViT برای طبقه بندی تصاویر گلبولهای سفید 10

2-1. مقدمه 10

2-2. مجموعه دادگان و آماده‌سازی 10

2-3. پیاده‌سازی مدل ViT 12

2-3-1. Classifier 12

2-3-2. دو لایه اول 13

2-3-3. دو لایه آخر انکودر 14

2-3-4. تمام لایه ها 16

2-4. پیاده‌سازی مدلCNN 17

2-4-1. تمام لایه ها 18

2-4-2. لایه دسته بند 19

2-5. تحلیل و نتیجه گیری 21

شکل‌ها

پرسش 1

شکل 1.1 :فرآیند کلی الگوریتم Slime Mould

شکل 2.1 :نمودار نیروی بادی فراساحلی و جداسازی داده به دو بخش train و test

شکل 3.1 :نمودار نیروی بادی فراساحلی و تشخیص داده‌های پرت

شکل 4.1 :نمودار نیروی بادی فراساحلی و دینویز کردن داده‌ها

شکل 5.1 :تقریب داده تست نیروی باد فراساحلی با 6 مدل خواسته‌شده برای t + 1

شکل 6.1 :تقریب داده تست نیروی باد فراساحلی با 6 مدل خواسته‌شده برای t + 4

شکل 7.1 :تقریب داده تست نیروی باد فراساحلی با 6 مدل خواسته‌شده برای t + 8

شکل 8.1 :تقریب داده تست نیروی باد فراساحلی با 6 مدل خواسته‌شده برای t + 16

پرسش 2

شکل 2-1 :نمایش کلاس‌های مختلف درBCCD

شکل 2-2 :تعداد دادگان هر کلاس

شکل 2-3 :تعداد دادگان هر کلاس بعد از تقویت داده

شکل 2-4 :انکودر درViT ، عکس از اسلایدهای درس

شکل 2-5 :نمودار آموزش با آزاد کردن لایهclassifier

شکل 2-6 :نمودار آزمون با آزاد کردن لایهclassifier

شکل 2-7 :نمودار آموزش با آزاد کردن دو لایه اول انکودر و دسته بند

شکل 2-8 :نمودار آزمون با آزاد کردن دو لایه اول انکودر و دسته بند

شکل 2-9 :نمودار آموزش با آزاد کردن دو لایه آخر رمزگذار و دسته بند

شکل 2-10 :نمودار آزمون با آزاد کردن دو لایه آخر رمزگذار و دسته بند

شکل 2-11 :نمودار آموزش با آزاد کردن همه لایه‌ها

شکل 2-12 :نمودار آزمون با آزاد کردن همه لایه‌ها

شکل 2-13 :ساختار شبکهDenseNet121

شکل 2-14 :نمودار آموزش با آزاد کردن همه لایه‌ها

شکل 2-15 :نمودار آزمون با آزاد کردن همه لایه‌ها

شکل 2-16 :نمودار آموزش با آزاد کردن لایه دسته بند

شکل 2-17 :نمودار آزمون با آزاد کردن لایه دسته بند

جدول‌ها

پرسش 1

جدول 1.1: نتایج بدست آمده از آموزش مدل‌ها برای $t + 1$

جدول 2.1: نتایج بدست آمده از آموزش مدل‌ها برای $t + 4$

جدول 3.1: نتایج بدست آمده از آموزش مدل‌ها برای $t + 8$

جدول 4.1: نتایج بدست آمده از آموزش مدل‌ها برای $t + 16$

پرسش 2

جدول 2-1: جمع‌بندی نتایج پرسش 2

پرسش ۱. پیش‌بینی نیروی باد به کمک مبدل و تابع خطای Huber

1-1. مقدمه

- روش‌های ARIMA و GARCH نیاز به مفروضات خاصی در مورد توزیع داده‌ها دارند. در نتیجه انعطاف پذیری آن ها در صورت اعمال مجموعه داده‌های که این معیارها را براورده نمی‌کند محدود است. همچنین، این مدل‌ها به نرمی داده‌ها بستگی دارند و ممکن است که با مجموعه داده‌های با نوسات بالا مبارزه کنند. این مدل‌ها تعمیم پذیری محدودی دارند و به دلیل محدودیت‌های ذاتی چارچوب‌های آماری آن‌ها محدود شده‌است.
- مدل‌های ماشینی به طور موثرتری با ساختار داده‌های متنوع سازگار می‌شود و در زمینه‌های متفاوتی به خوبی عمل می‌کنند. همچنین، این مدل‌ها وابستگی کمتری به پیش‌پردازش داده‌ها دارند. این مدل‌ها نیز، با نوسانات و پیچیدگی بالا سازگاری بهتری دارند.
- مدل‌های یادگیری عمیق، می‌توانند روابط زمانی پیچیده را نیز یاد بگیرند. این مدل‌ها، هیچ گونه فرضیاتی در مورد توزیع داده‌ها ندارند. همچنین، این مدل‌ها تعمیم بهتری در زمینه‌های مختلف دارند. همچنین، این مدل‌ها میتوانند چالش‌های غیرخطی را کنترل کنند.
- مکانیسم خودتوجهی می‌تواند الگوهای کوتاه‌مدت یا محلی را با داشتن توجه به وسیله وزن‌دهی به المان‌های نزدیک کنترل کنند. این مدل، نسبت به نویز استحکام بیشتری دارد. همچنین، این مدل می‌تواند همبستگی‌های پیچیده را در توالی‌های با مقیاس‌های متفاوت استخراج کند.
- این تابع overfitting نسبت به ناهنجاری‌ها را کاهش می‌دهد و با مدیریت داده‌های پرت، از یادگیری این نقاط جلوگیری می‌کند. همچنین به دلیل ماهیت تطبیقی این تابع، مدل دقت را برای خطاهای کوچک و استحکام را برای خطاهای بزرگ متعادل می‌کند. این تابع نیز تعمیم‌پذیری بیشتری در مورد داده‌های فصلی دارد.

1-2. آماده‌سازی

- ساختار Autoencoder، یک پارادایم یادگیری بدون نظارت است. این شبکه عصبی برای یادگیری یک نمایش فشرده و کم‌بعد از داده‌ها طراحی شده‌است. این شبکه به هدف حذف نویز، استخراج ویژگی‌ها، کاهش ابعاد و بازسازی داده‌ها استفاده می‌شود. یک Autoencoder از دو بخش اصلی emcoder و decoder ساخته شده‌است. بخش encoder داده ورودی را به یک فضای با نمایش نهفته و فشرده می‌برد. بخش decoder نیز از نمایش نهفته داده را به حالت

اول خود باز می‌گرداند. هدف این بخش این است که داده بازسازی شده شباهت بالایی با داده اصلی داشته باشد.

- در transformer ها، مکانیزم توجه برای پردازش داده‌های ترتیبی طراحی شده‌است و دو هدف درک وابستگی‌های بلندمدت و وزن‌دهی پویا را دارد. Position Encoding نیز به هدف حفظ ترتیب داده‌ها و بهبود درک روابط زمانی یا مکانی تعبیه شده‌است. Position Encoding به نحوه زیر محاسبه می‌شود:

$$PE(pos, 2i) = \sin\left(pos/10000^{2i/d}\right)$$
$$PE(pos, 2i+1) = \cos\left(pos/10000^{2i/d}\right)$$

- تابع خطای Huber ترکیبی از دو تابع Squared Loss و Absolute Loss می‌باشد و ویژگی این دو تابع خطا را با یکدیگر ترکیب می‌کند. این تابع برای داده‌های شامل مقادیر پرت و با نویز طراحی شده‌است. شکل ریاضی این تابع خطا به صورت زیر است:

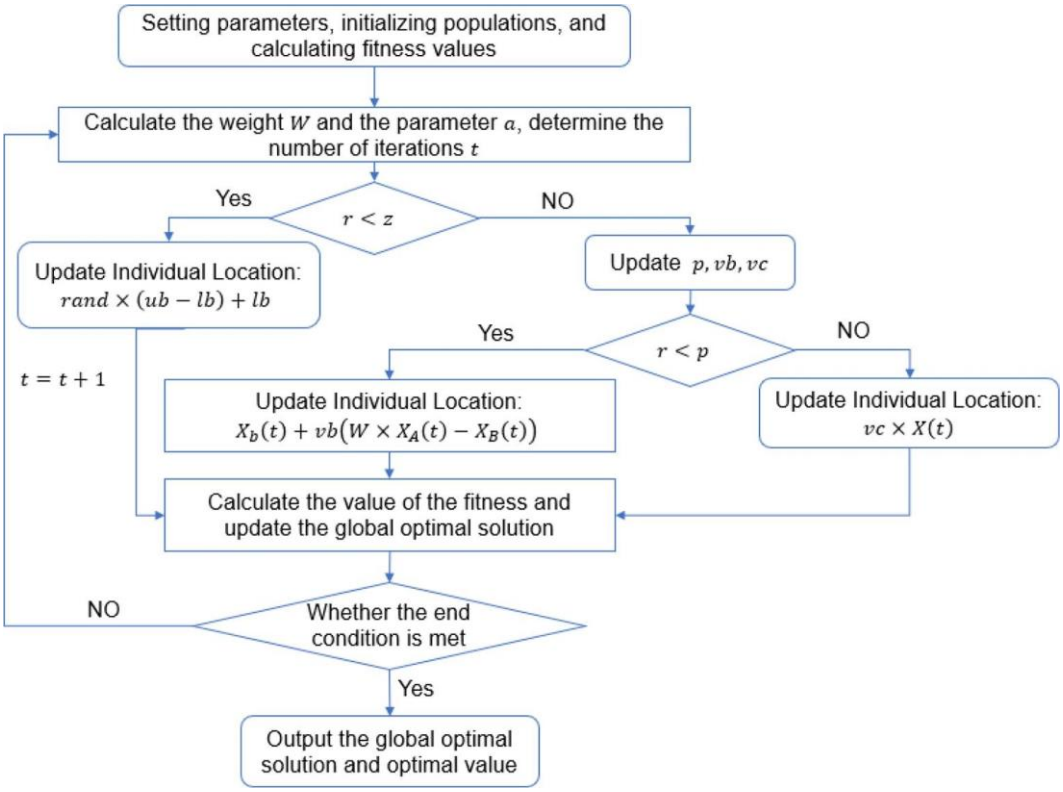
$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & |y-\hat{y}| \le \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta^2 & otherwise \end{cases}$$

در این تعریف، δ یک هایپرپارامتر می‌باشد. مشاهده می‌شود که این تابع برای خطاهای کوچک مانند تابع Squared Loss رفتار می‌کند. در این بازه، تمرکز تابع بر کاهش خطاهای کوچک‌تر و ایجاد مدل دقیق‌تر است. برای خطاهای بزرگ رفتار این تابع مانند Absolute Loss می‌شود و تمرکز بر جلوگیری از تاثیر زیاد مقادیر پرت در فرآیند آموزش است. داده نیروی باد شامل نویز و نوسانات زیاد می‌باشد؛ استفاده از تابع خطای Huber باعث می‌شود که مدل به خطاهای کوچک حساس باشد و همچنین در برابر مقادیر پرت نیز مقاومت بالایی کسب کند. همچنین، فرآیند بهینه‌سازی به دلیل مشتق‌پذیری پیوسته تابع Huber ثبات بیشتری دارد.

- الگوریتم Slime mould از رفتار جست و جوی مواد غذایی قالب‌های مخاطی الهام گرفته شده‌است. این موجودات با ترکیب انتشار شیمیایی و حرکت سلولی به جست و جوی مواد مغذی می‌پردازند و از مواد مضر نیز اجتناب می‌ورزند. مراحل اصلی این الگوریتم به ترتیب نزدیک شدن به منابع غذایی، محاصره منابع و کسب مواد غذای می‌باشد. این رفتار به صورت ریاضی نیز پیاده‌سازی شده‌است:

$$X(t+1) = \begin{cases} rand \times (ub - lb) + lb & r < z \\ X_b(t) + VB\left(W \times X_A(t) - X_B(t)\right) & r < p \\ vc \times X(t) & r \ge p \end{cases}$$
$$p = \tanh(|S(i) - DF|)$$

شکل 1.1 نیز فرآیند کلی این الگوریتم را نشان می‌دهد.

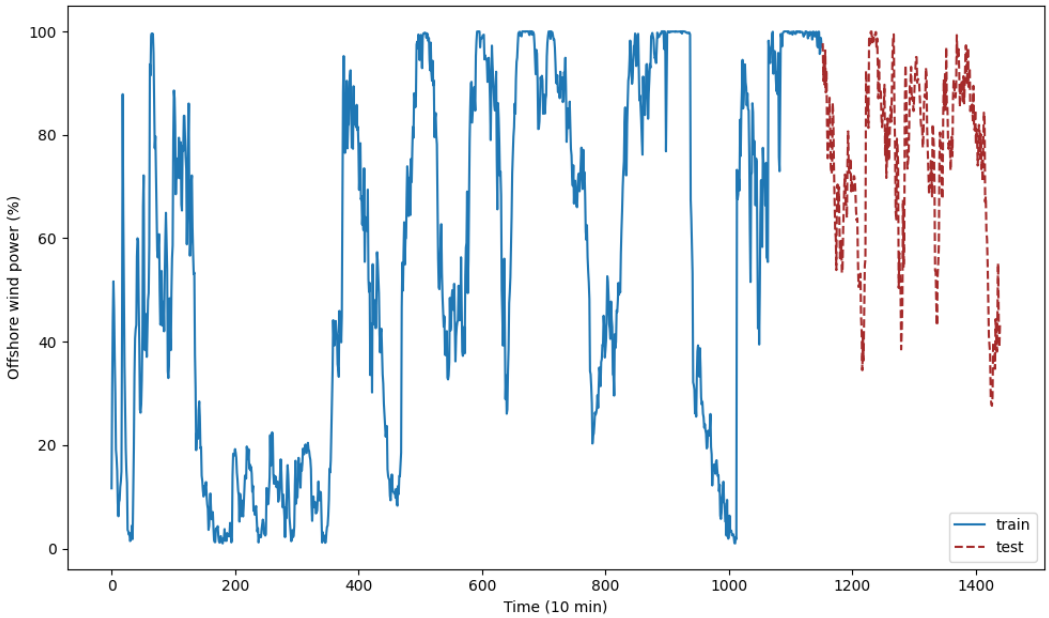


Slime Mould *شکل 1.1. فرآیند کلی الگوریتم*

تابع Slime Mould ورودی‌هایی از قبیل Objective Function (تابع هدف)، Dimensionality (ابعاد فضای جست و جو)، Maximum Iterations (بیشترین تعداد اجرای الگوریتم)، Bounds (حد بالا و پایین متغیر تصمیم) را نام برد. برای این تابع می‌تواند مقادیر دیگری چون delta و یا شرایطی برای توقف زودهنگام تعبیه کرد.

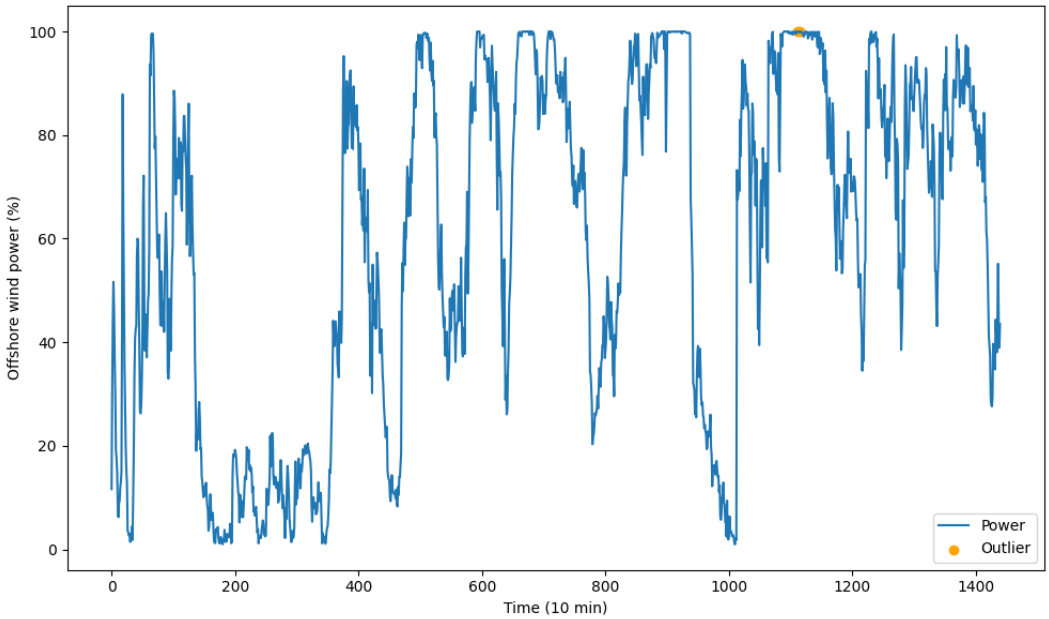
3-1. روش‌شناسی و نتایج

- با توجه به مقاله، یک روز از داده را به عنوان داده هدف انتخاب می‌کنیم. سپس، طبق مقاله 80% ابتدایی داده را به عنوان داده train و 20% باقی‌مانده داده را به عنوان داده test انتخاب می‌کنیم. با توجه به مطالب بالا، شکل 2.1 نمودار مطلوب را نشان می‌دهد.



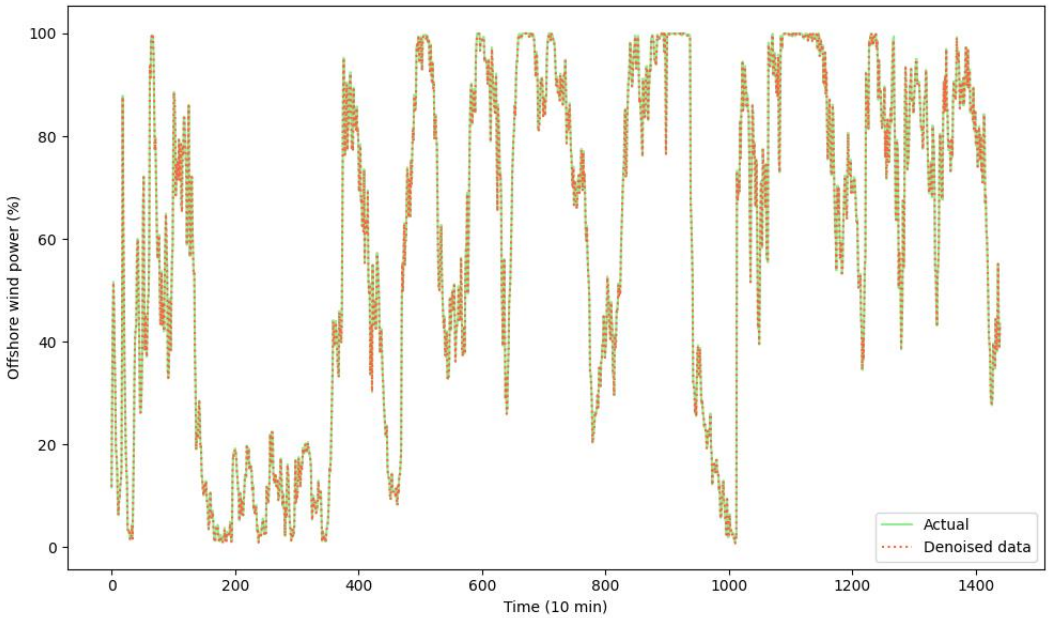
شکل 2.1. نمودار نیروی بادی فراساحلی و جداسازی داده به دو بخش ***train*** *و* ***test***

- مجددا طبق مقاله به تشخیص داده‌های پرت می‌پردازیم. این داده‌ها برای هر پنجره از رابطه $|X_i - \mu_{i,w}| > 2 \times \sigma_{i,w}$ بدست می‌آید. w به پنجره گفته شده می‌باشد. شکل 3.1 نمودار حاصل شده را نشان می‌دهد.



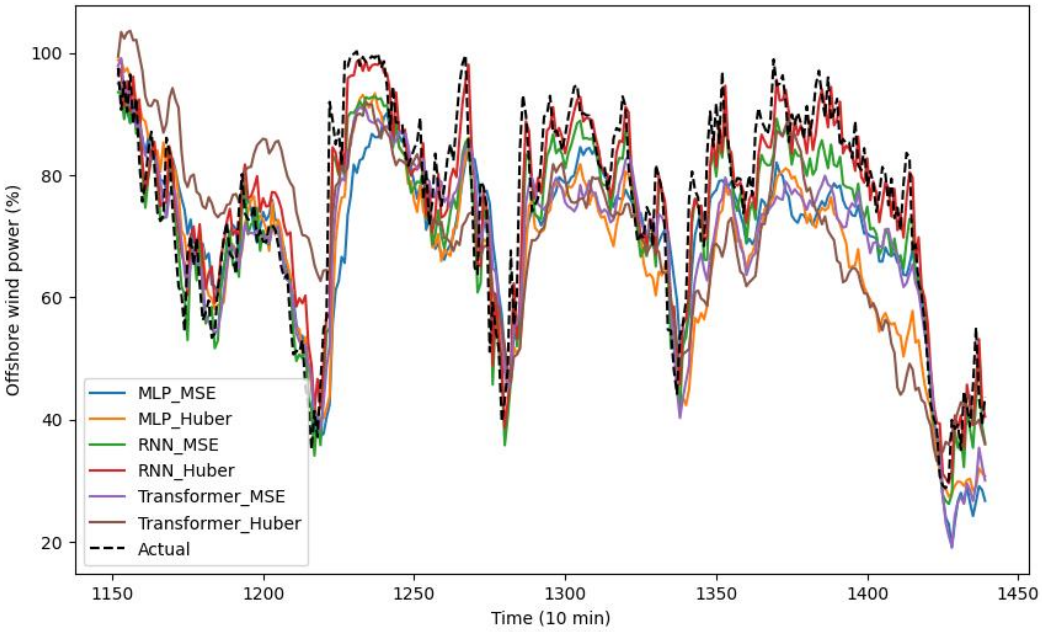
شکل 3.1. نمودار نیروی بادی فراساحلی و تشخیص داده‌های پرت

- به کمک اتوانکودر طراحی شده، اقدام به دینویز داده‌ها می‌کنیم. شکل 4.1 مقایسه داده اصلی با داده دینویز شده را نشان می‌دهد.



شکل 4.1. نمودار نیروی بادی فراساحلی و دینویز کردن داده‌ها

- برای پیش‌پردازش، ابتدا داده‌ها را طبق مقاله به دو دسته train و test تقسیم می‌کنیم. سپس با روش min-max normalization داده را نرمالیزه می‌کنیم و در انتها اقدام به پنجره پنجره کردن داده می‌کنیم. دلیل انجام جداسازی قبل جزو مفادی بود که در کلاس مطرح شد. در کلاس گفته‌شد که داده تست باید مطابق پارامترهای داده train آموزش ببیند؛ در نتیجه ابتدا باید جداسازی انجام دهیم و در ادامه داده را نرمالیزه کنیم. در نتیجه اگر ابتدا داده را نرمالیزه کنیم این قاعده رعایت نخواهد شد. به عنوان مثال نقض، فرض کنید ابتدا داده‌ها نرمال‌سازی شوند و سپس تقسیم به مجموعه آموزش و تست انجام گیرد. در این حالت، مقادیر نرمال‌سازی شده مجموعه تست بر اساس کل داده‌ها محاسبه شده‌است، که باعث می‌شود مدل به اطلاعات آینده دسترسی داشته باشد و عملکرد واقعی مدل به‌درستی ارزیابی نشود.
- طبق خواسته سوال، 6 مدل را به تعداد 50 ایپاک با پارامترهای گفته‌شده آموزش می‌دهیم. سپس، داده test را روی این 6 مدل بررسی می‌کنیم؛ بعد از آن، اقدام به بازنشانی نرمالیزیشن داده تست می‌کنیم. شکل 4.1 این نتایج را نشان می‌دهد.



*شکل 5.1. تقریب داده تست نیروی باد فراساحلی با **6** مدل خواسته‌شده برای **t + 1***

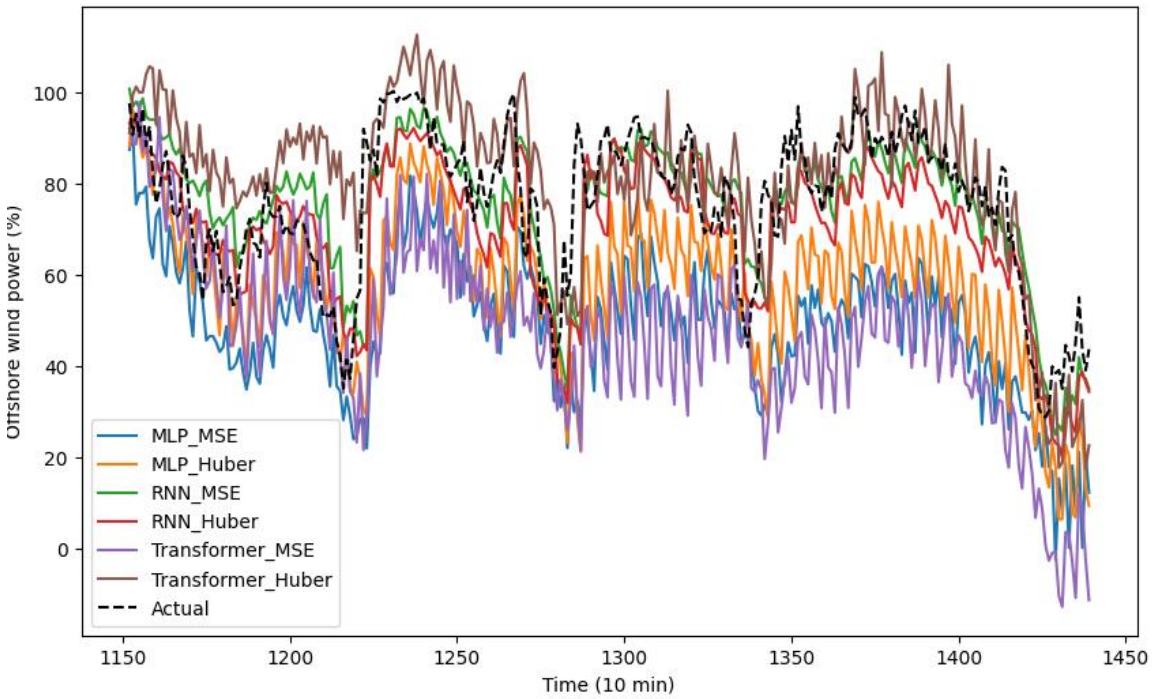
پس از بررسی نمودار مشاهده می‌کنیم که عملکرد ترانسفورمرها از سایر مدل‌ها بهتر است، همچنین، تابع هزینه Huber، به مراتب بهتر از MSE می‌باشد.

- جدول 1.1 نتایج خواسته‌شده این قسمت را نشان داده‌است.

Index	*Model*	*MSE*	*Huber*
MAE	MLP	0.041744	0.041905
	RNN	0.040942	0.041595
	Transformer	0.053986	0.048387
MAPE	MLP	5.956366	10.399977
	RNN	6.231303	1.551221
	Transformer	6.438055	8.807488
RMSE	MLP	0.057559	0.059330
	RNN	0.062102	0.061303
	Transformer	0.072309	0.063830

جدول 1.1. نتایج بدست آمده از آموزش مدل‌ها

- برای قسمت امتیازی، پارامتر نرخ یادگیری را به عنوان پارمتر هدف در نظر می‌گیریم و این را به صورت یک objective_function به تابع slime_mould می‌دهیم.
- برای پیش‌بینی multi-step نتایج زیر حاصل می‌شود:
  - برای t + 4 نتایج زیر حاصل می‌شود.



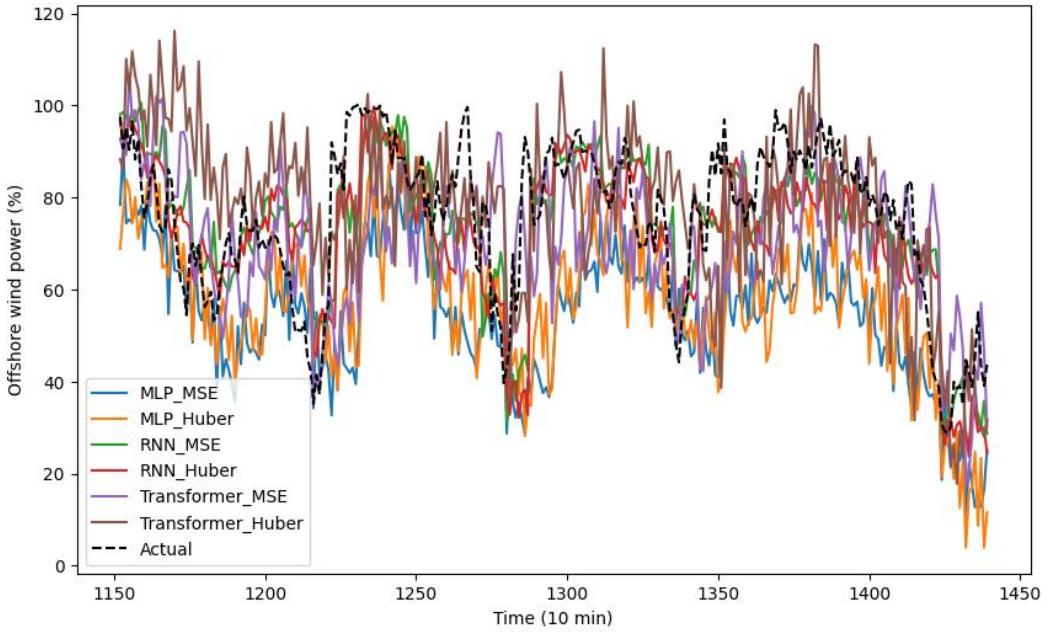
شکل 6.1. تقریب داده تست نیروی باد فراساحلی با ***6*** *مدل خواسته‌شده برای* ***t + 4***

دلیل نویزی بودن داده‌ها، عملکرد نه‌چندان مناسب اتوانکودر می‌باشد. این انکودر مقدار قابل توجهی داده را دینویز نمی‌کند؛ در نتیجه نویز در داده پیش‌بینی شده مشهود است، اما مدل‌ها به درستی پوش سیگنال را دنبال می‌کنند.

Index	*Model*	*MSE*	*Huber*
MAE	MLP	0.093367	0.069099
	RNN	0.062453	0.067038
	Transformer	0.080162	0.067648
MAPE	MLP	22.217205	37.607859
	RNN	28.237775	23.173198
	Transformer	13.620536	25.397750
RMSE	MLP	0.119811	0.091921
	RNN	0.094069	0.099649
	Transformer	0.103653	0.087330

جدول 2.1. نتایج بدست آمده از آموزش مدل‌ها برای **t + 4**

- برای t + 8 نتایج زیر حاصل می‌شود.



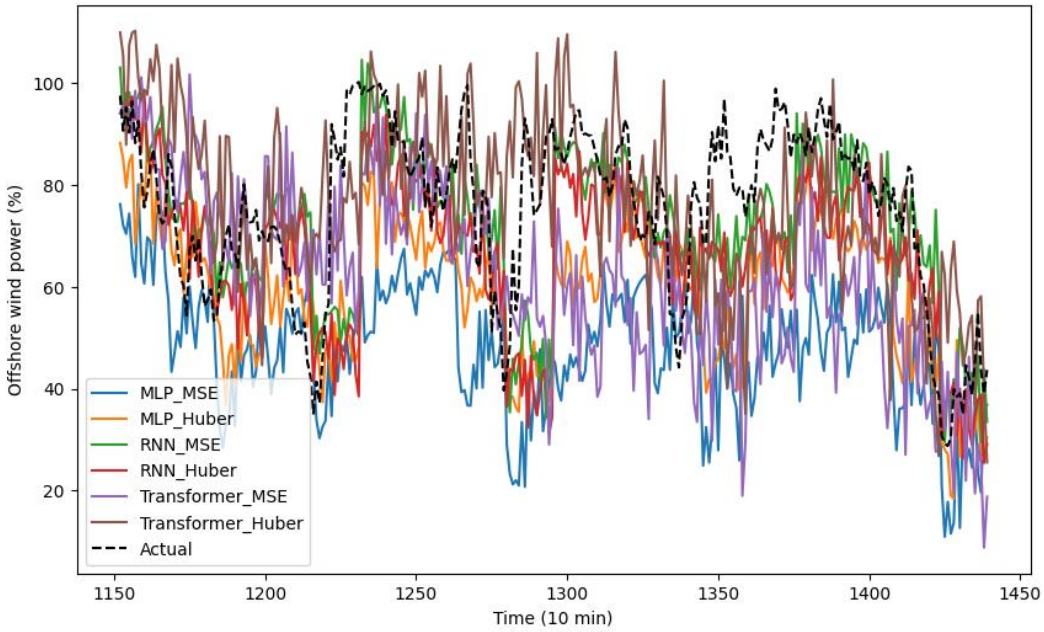
شکل 7.1. تقریب داده تست نیروی باد فراساحلی با **6** *مدل خواسته‌شده برای* **t + 8**

دلیل نویزی بودن داده‌ها، عملکرد نه‌چندان مناسب اتوانکودر می‌باشد. این انکودر مقدار قابل توجهی داده را دینویز نمی‌کند؛ در نتیجه نویز در داده پیش‌بینی شده مشهود است، اما مدل‌ها به درستی پوش سیگنال را دنبال می‌کنند.

Index	*Model*	*MSE*	*Huber*
MAE	MLP	0.101446	0.094266
	RNN	0.083746	0.080241
	Transformer	0.084633	0.087689
MAPE	MLP	39.620337	24.744039
	RNN	63.023063	50.989587
	Transformer	54.103031	45.640831
RMSE	MLP	0.135167	0.128540
	RNN	0.119638	0.113844
	Transformer	0.108101	0.110085

جدول 3.1. نتایج بدست آمده از آموزش مدل‌ها برای **t + 8**

- برای t + 16 نتایج زیر حاصل می‌شود.



شکل 8.1. تقریب داده تست نیروی باد فراساحلی با **6** مدل خواسته‌شده برای **t + 16**

دلیل نویزی بودن داده‌ها، عملکرد نه‌چندان مناسب اتوانکودر می‌باشد. این انکودر مقدار قابل توجهی داده را دینویز نمی‌کند؛ در نتیجه نویز در داده پیش‌بینی شده مشهود است، اما مدل‌ها به درستی پوش سیگنال را دنبال می‌کنند.

Index	*Model*	*MSE*	*Huber*
MAE	MLP	0.139649	0.135072
	RNN	0.128350	0.127774
	Transformer	0.092818	0.092020
MAPE	MLP	205.882262	184.605112
	RNN	228.473100	233.245550
	Transformer	115.626150	74.634631
RMSE	MLP	0.178896	0.170270
	RNN	0.173478	0.173530
	Transformer	0.119114	0.116872

جدول 4.1. نتایج بدست آمده از آموزش مدل‌ها برای **t + 8**

با توجه به اینکه فرکانس ثبت داده 10 min می‌باشد، مدل اول می‌توانست 10 min بعد، مدل دوم می‌توانست 40 min بعد، مدل سوم 80 min بعد و مدل آخر 160 min بعد را پیش‌بینی می‌کرد.

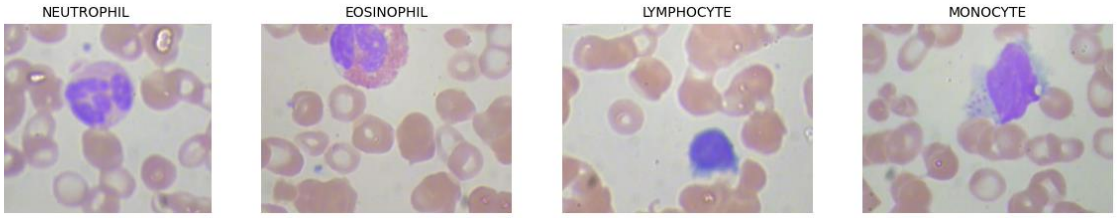
پرسش 2 – استفاده از ViT برای طبقه بندی تصاویر گلبولهای سفید

2-1. مقدمه

برای این پرسش، داده های BCCD که به 4 دسته تقسیم شده بودند استفاده شد. مدل ها در محیط colab و با استفاده از T1 GPU آموزش داده شدند.

2-2. مجموعه دادگان و آماده‌سازی

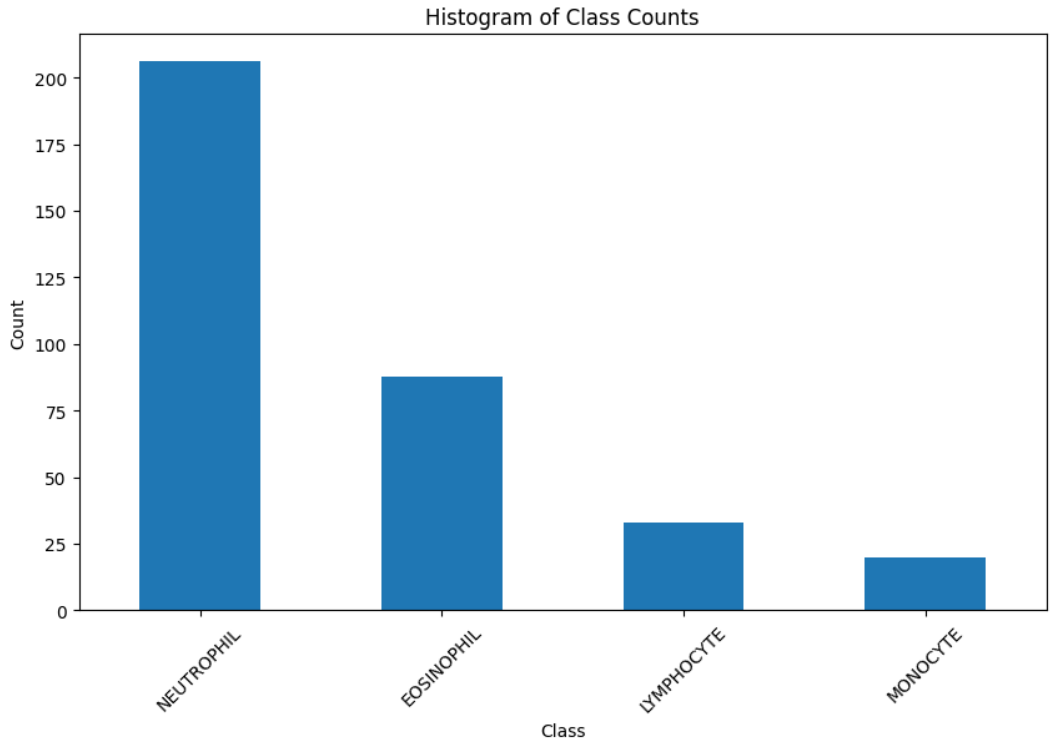
ابتدا دیتاست مورد نظر را توسط کتابخانه pandas لود میکنیم و با استفاده از pyplotlib نشان میدهیم. توجه شود که به علت ساختار کد، مرحله نمایش کلاس ها (که در شکل 2-1 قابل مشاهده است) بعد از عمل augmentation انجام داده شده است.



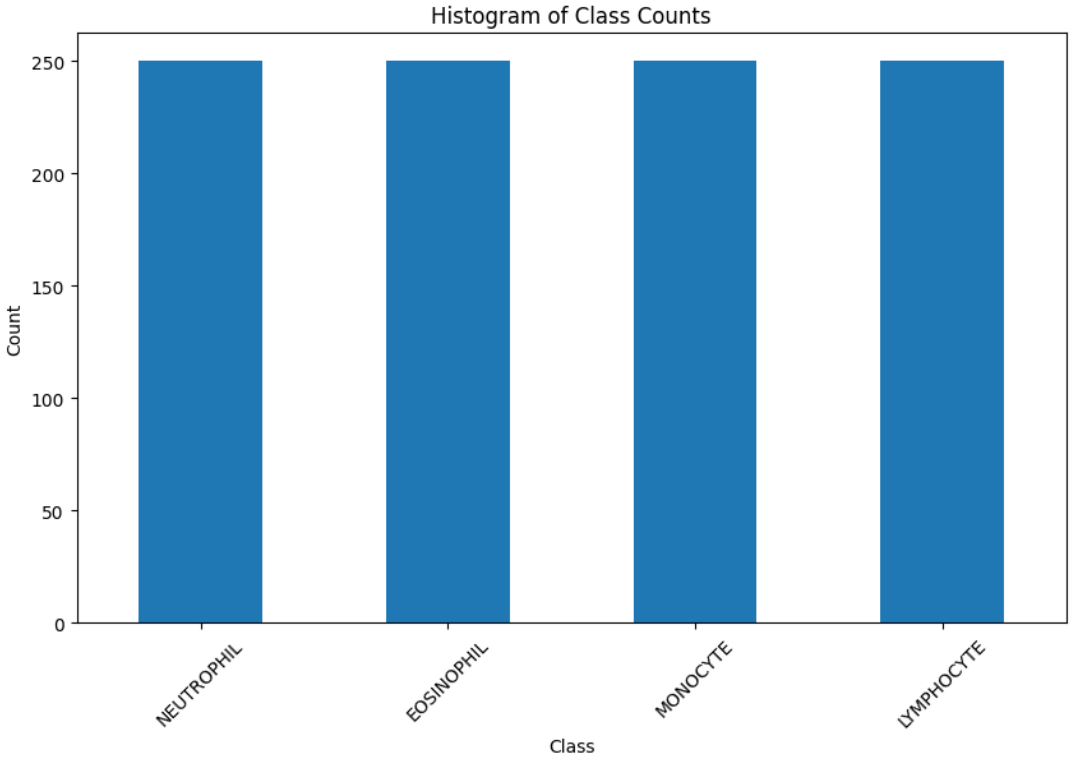
شکل 2-1. نمایش کلاس های مختلف در ***BCCD***

با مقایسه تعداد عکس های موجود به ازای هر کلاس، متوجه میشویم که داده ها به شدت نامتوازن هستند و باید کلاس ها را تقویت کنیم. تعداد داده های موجود در هر کلاس در شکل 2-2 نشان داده شده است.

به منظور تقویت داده های هر کلاس، از کتابخانه albumentations استفاده شده است. این کتابخانه توابعی برای انجام augmentation و همچنین تبدیل annotation ها برای داده های جدید دارد. برای تقویت از 4 عمل آیینه ای کردن عکس به صورت عمودی و افقی، چرخش عکس و تغییر روشنایی عکس استفاده شده است که به طور رندم اعمال میشوند. میزان تقویت طوری تنظیم شده که بعد از آن از هر کلاس 250 نمونه و در مجموع 1000 داده برای آموزش و آزمون داشته باشیم. داده های تقویت شده بعد از اینکه به دو دسته ی آزمون و آموزش (به نسبت 1 به 9) تقسیم شدند، به دیتاست pytorch تبدیل میشوند و به سایز 224x224 که سایز مطلوب برای ViT مورد استفاده است در میایند. در نهایت برای تقسیم داده ها به batch های مختلف از دیتالودر استفاده میکنیم و میزان هر دسته را 16 قرار میدهیم.



شکل 2-2. تعداد دادگان هر کلاس

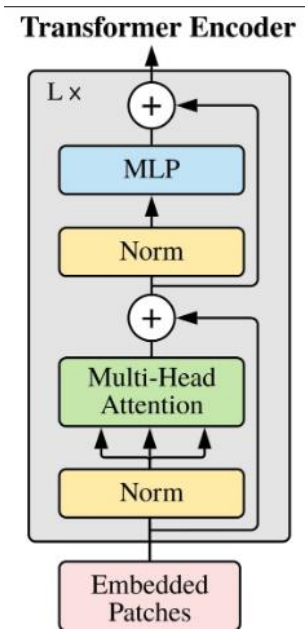


شکل 2-3. تعداد دادگان هر کلاس بعد از تقویت داده

2-3. پیاده‌سازی مدل ViT

برای ترنسفورمر از مدل vit-base-patch16-224 گوگل استفاده کردیم و تعداد کلاس های آن را برابر 4 قرار دادیم. این مدل از ساختار ViT استفاده میکند و برای تشخیص کلاس های مربوط به اشیا به کار میرود. مدل بر روی ImageNet آموزش داده شده است که دارای 1000 کلاس مختلف برای اشیا است و مدل پیش فرض به همین تعداد خروجی دارد. ورودی مدل به patch های 16 در 16 تقسیم میشود و از این پیکسل ها به عنوان ورودی به جای توکن ها استفاده میکند. ابعاد ورودی پیشفرض این مدل، همانطور که قبلا هم اشاره شد 224 در 224 است. با خروجی گرفتن از شکل مدل، متوجه میشویم که این مدل از لایه های زیر تشکیل شده است:

- لایه Embeddings که patch های مورد بحث در این لایه تولید میشوند (علت این نامگذاری آن است که ترنسفورمر ها برای داده های متنی طراحی شده اند و برای استفاده از آن در عکس باید عکس را به فرمتی شبیه به متن تبدیل کنیم).
- لایه انکودر: بخش عمده ای از یادگیری در این بخش اتفاق می افتد. خود این لایه از 12 زیر لایه کوچکتر تشکیل شده که شامل لایه های attention, linear, dropout,dense و اکتیویشنGELU میباشند. ساختار این لایه مشابه شکل 2-4 میباشد.
- لایه آخر این مدل یک لایه classifier است که تقسیم بندی نهایی را انجام میدهد.

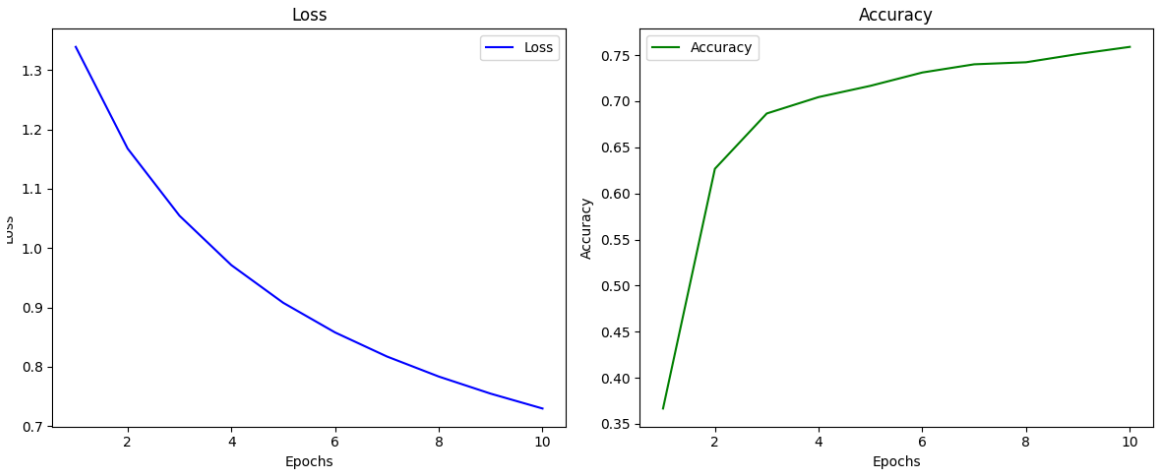


*شکل 2-4. انکودر در **ViT**، عکس از اسلاید های درس*

برای آموزش تمام مراحل از 10 epoch و ضریب آموزش 10^-4 استفاده شده است و از Adam به عنوان اپتیمایزر استفاده کردیم.

2-3-1. Classifier

مدل اصلی 85801732 پارامتر دارد که 3076 تا مربوط به لایه classifier میباشند. برای این مرحله از سوال، تنها این لایه را قابل ترین قرار داده و بقیه لایه ها را فیکس میکنیم. این راهکار نسبت به قسمت های بعدی سریع تر است و احتمال overfit کمتری دارد، ولی به توجه به اینکه تعداد پارامتر های آموزش داده شده بسیار کمتر از کل پارامترهاست (کمتر از 0.004 درصد کل) نتایج دقت بالایی ندارند. نتیجه آموزش در شکل 2-5 و دقت تست در شکل 2-6 نمایش داده شده اند.



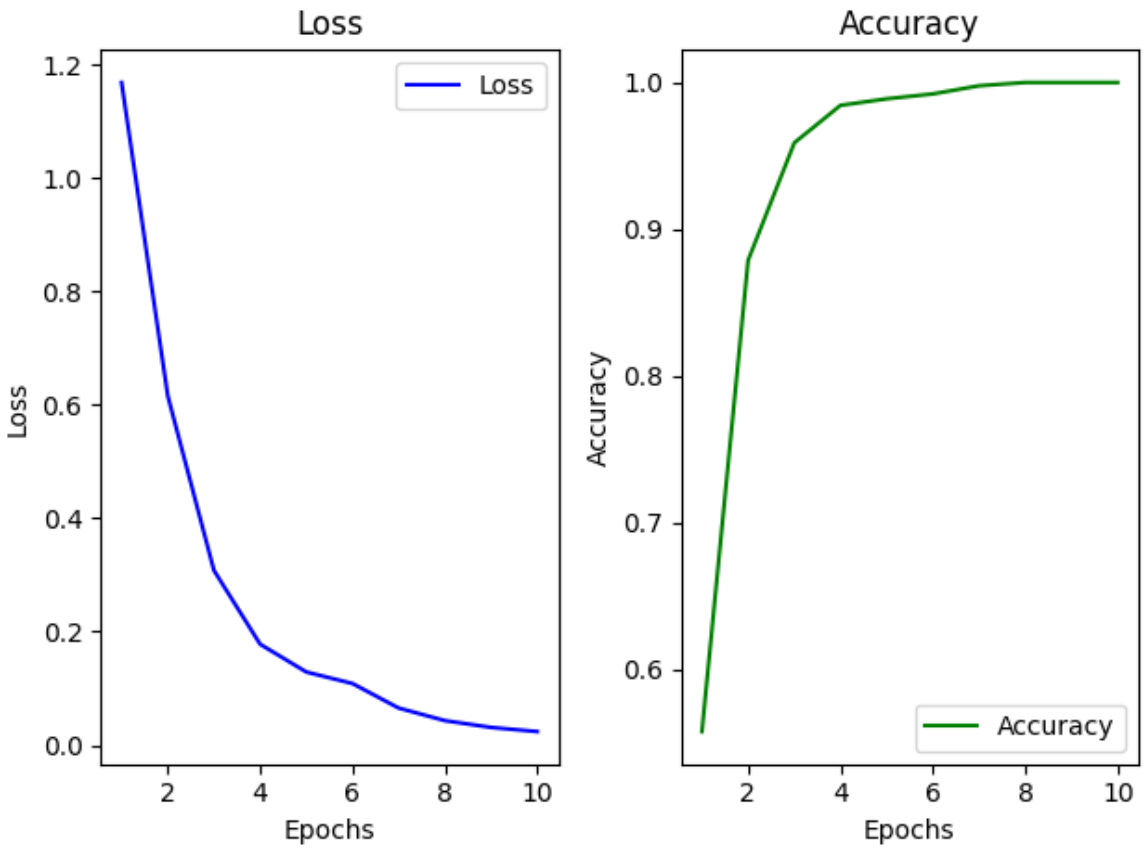
شکل 2-5. نمودار آموزش با آزاد کردن لایه ***classifier***

	precision	recall	f1-score	support
NEUTROPHIL	0.77	0.80	0.78	25
EOSINOPHIL	0.74	0.56	0.64	25
MONOCYTE	0.71	0.80	0.75	25
LYMPHOCYTE	0.81	0.88	0.85	25
accuracy			0.76	100
macro avg	0.76	0.76	0.76	100
weighted avg	0.76	0.76	0.76	100

شکل 2-6. نمودار آزمون با آزاد کردن لایه ***classifier***

2-3-2. دو لایه اول

مدل اصلی 85801732 پارامتر دارد که 14178820 تا مربوط به دو لایه اول انکودر و دسته بند میباشند. این تعداد حدود 16.5 درصد از کل پارامتر ها را تشکیل میدهد و نسبت به راهکرد قبلی به مراتب بیشتر است. در این مرحله از لحاظ دقت به نتیجه ی بهتری میرسیم و سرعت همگرایی نیز بیشتر است، اما در ازای آن سرعت آموزش بیشتر میشود و هزینه ی محاسباتی بیشتر است. نتایج این مرحله در تصاویر 2-7 و 2-8 قابل مشاهده است.



شکل 2-7. نمودار آموزش با آزاد کردن دو لایه اول انکودر و دسته بند

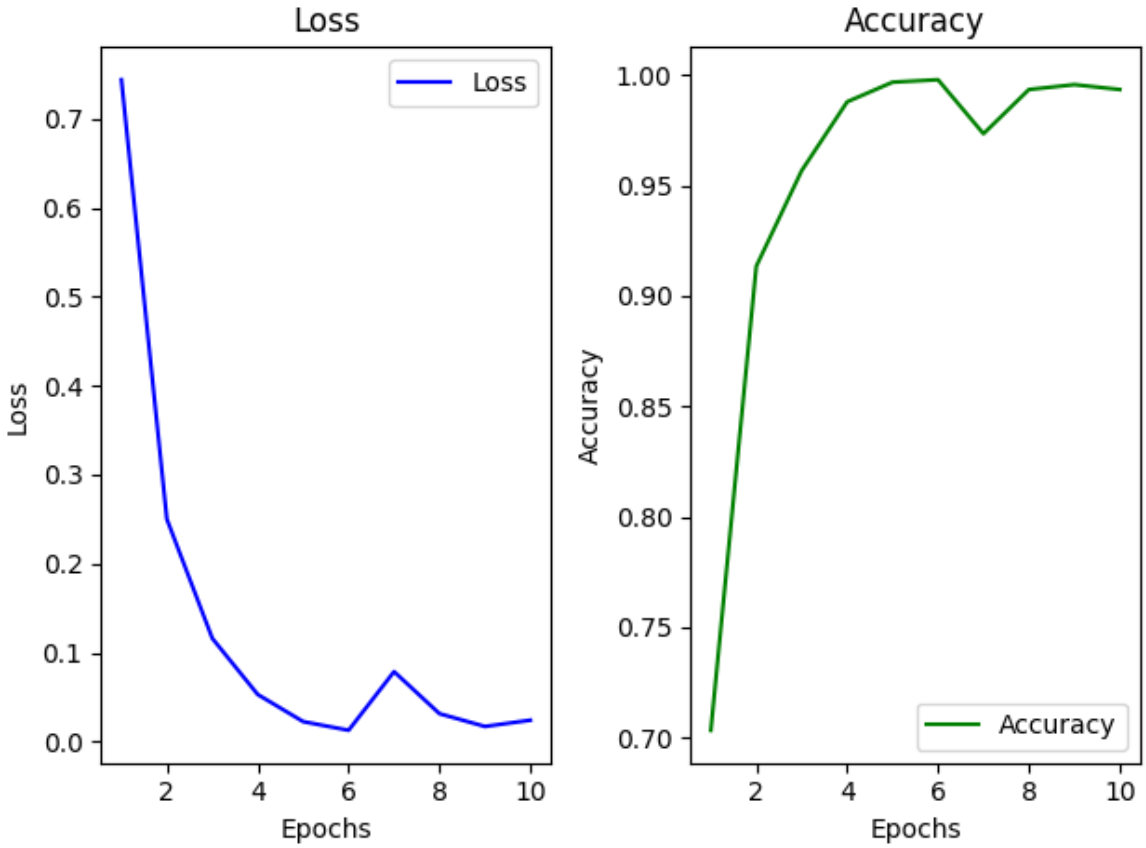
	precision	recall	f1-score	support
NEUTROPHIL	0.95	0.84	0.89	25
EOSINOPHIL	0.86	0.96	0.91	25
MONOCYTE	1.00	0.96	0.98	25
LYMPHOCYTE	0.96	1.00	0.98	25
accuracy			0.94	100
macro avg	0.94	0.94	0.94	100
weighted avg	0.94	0.94	0.94	100

شکل 2-8. نمودار آزمون با آزاد کردن دو لایه اول انکودر و دسته بند

2-3-3. دو لایه آخر انکودر

در این مرحله تعداد پارامتر ها با حالت قبلی یکی است و تنها محل قرار گیری پارامتر های ترین شده متفاوت است. در این حالت چون پارامتر ها در لایه های بالاتری قرار دارند، با داده های سطح بالا و نزدیکتر به دسته بندی نهایی کار میکنند و تاثیر آن در نتیجه ی آموزش بیشتر است. با مقایسه نتایج موجود در

اشکال 2-9 و 2-10 با نتایج مرحله قبلی متوجه میشویم که این راه حل در تمام معیار ها حدود 3 درصد بهتر عمل کرده و لایه های بالاتر تاثیر بیشتری بر روی نتیجه ی مدل دارند.



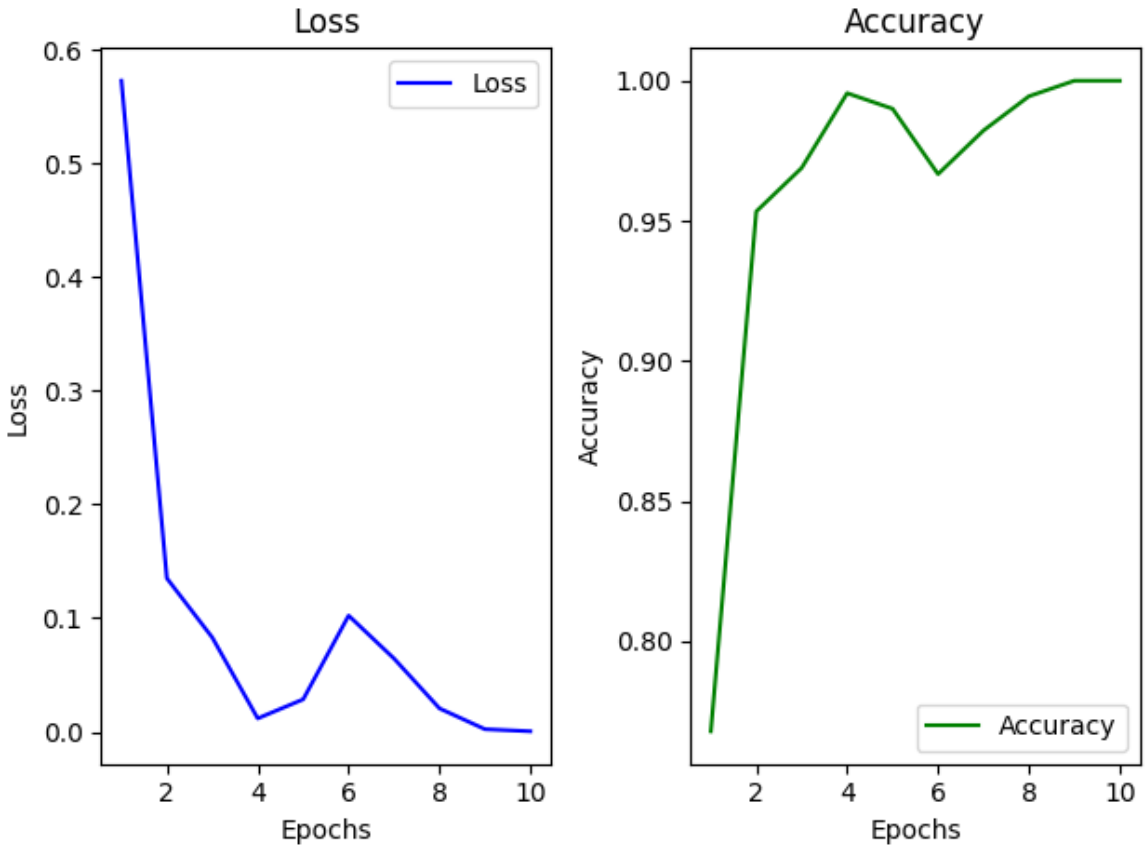
شکل 2-9. نمودار آموزش با آزاد کردن دو لایه آخر رمزگذار و دسته بند

	precision	recall	f1-score	support
NEUTROPHIL	0.96	0.92	0.94	25
EOSINOPHIL	0.92	0.96	0.94	25
MONOCYTE	1.00	1.00	1.00	25
LYMPHOCYTE	1.00	1.00	1.00	25
accuracy			0.97	100
macro avg	0.97	0.97	0.97	100
weighted avg	0.97	0.97	0.97	100

شکل 2-10. نمودار آزمون با آزاد کردن دو لایه آخر رمزگذار و دسته بند

2-3-4. تمام لایه ها

در این مرحله تمامی لایه ها را باز میکنیم و کل مدل را ترین میکنیم. این روش بین حالت های مختلف بیشترین هزینه زمانی را دارد و نتایجی که میگیریم (نمایش داده شده در شکل های 2-11 و 2-12) تقریبا با مرحله قبلی یکی است. همچنین نمودار خطا نشان میدهد که تعداد eopoch ها مقداری زیاد است و اگر این روند را ادامه بدهیم overfit میشود. بنابرین استفاده از روش قبلی منطقیتر است.



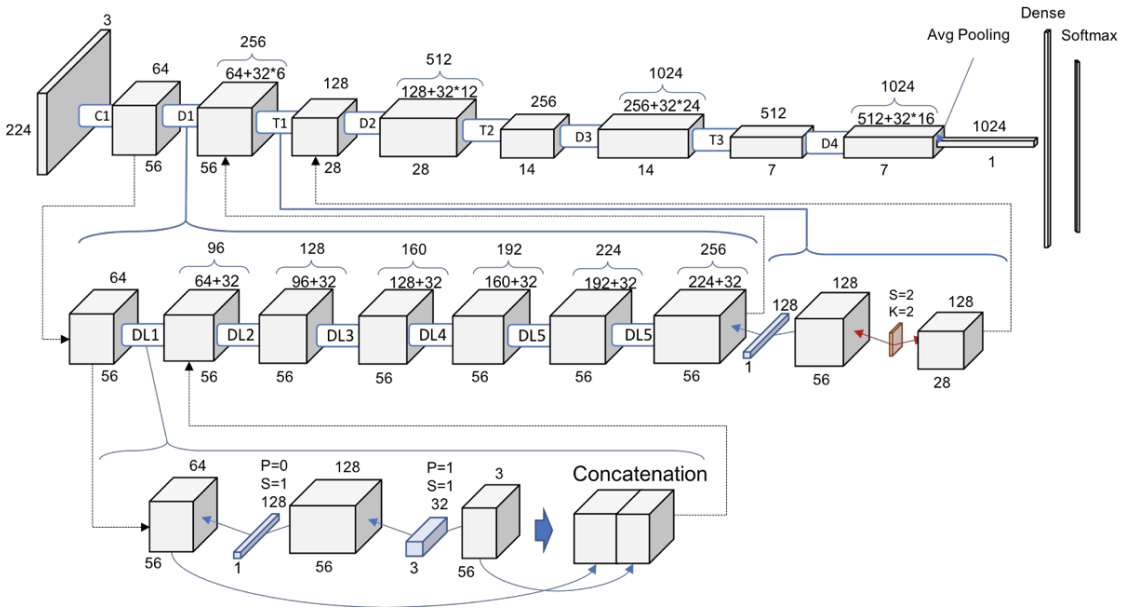
شکل 2-11. نمودار آموزش با آزاد کردن همه لایه ها

	precision	recall	f1-score	support
NEUTROPHIL	0.89	1.00	0.94	25
EOSINOPHIL	1.00	0.88	0.94	25
MONOCYTE	1.00	1.00	1.00	25
LYMPHOCYTE	1.00	1.00	1.00	25
accuracy			0.97	100
macro avg	0.97	0.97	0.97	100
weighted avg	0.97	0.97	0.97	100

شکل 2-12. نمودار آزمون با آزاد کردن همه لایه ها

2-4. پیاده‌سازی مدل CNN

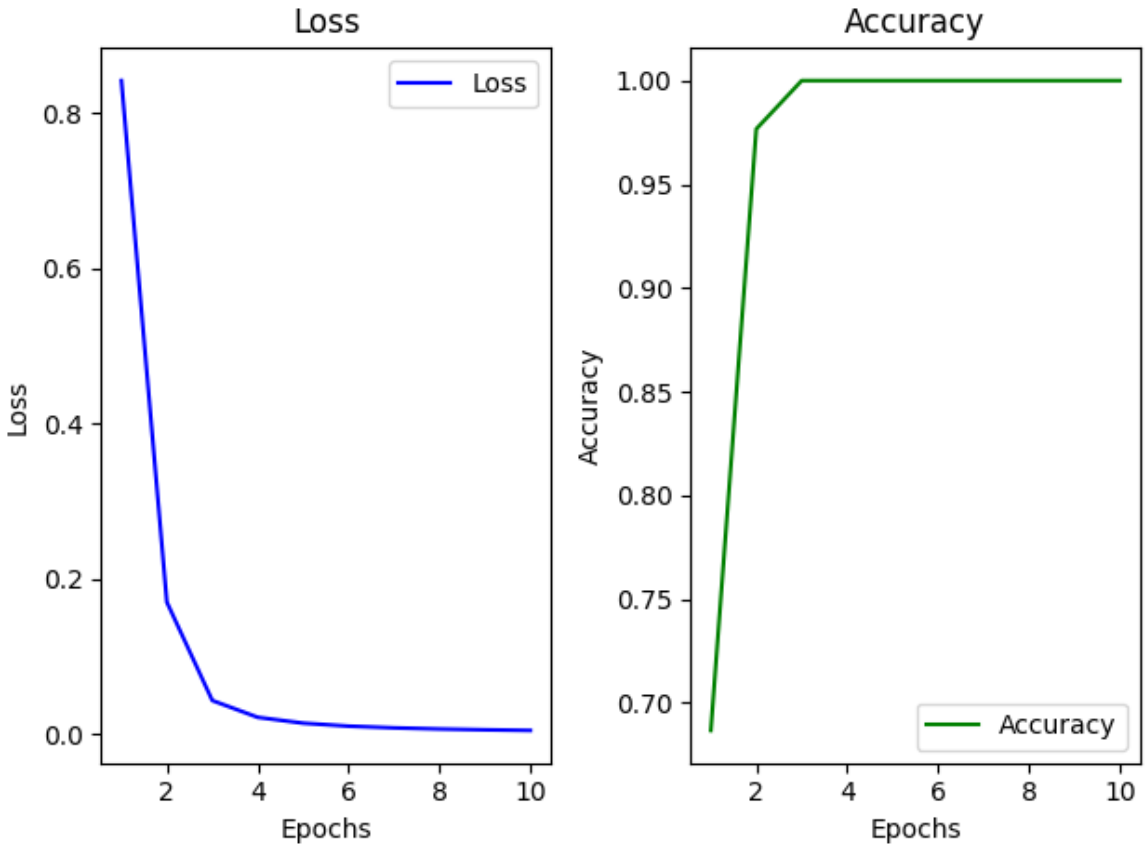
در این مرحله از مدل DenseNet121 که یک مدل CNN است استفاده شده است. برای بارگذاری این مدل از کتابخانه torchvision استفاده کردیم. این شبکه از لایه های متعدد convolution استفاده میکند و لایه ها را با استفاده از denseblock ها به همدیگر وصل میکند. این شبکه به علت بهینه بودن و سرعت بالا مورد استفاده است. برای تبدیل این مدل به حالتی که دسته بندی را با 4 کلاس انجام دهد، لایه دسته بند آن را با لایه جدیدی جایگزین کردیم که 4 نود خروجی دارد.



شکل 2-13. ساختار شبکه ***DenseNet121***

2-4-1. تمام لایه ها

در این مرحله تمامی لایه ها را باز میکنیم و کل مدل را ترین میکنیم. شبکه CNN در کل 6957956 پارامتر دارد که همه ی آنها در وضعیت آزاد قرار دارند. برای اندازه گیری خطا از تابع CrossEntropyLoss کتابخانه numpy استفاده کردیم. این مدل را نیز مانند مدل قبلی در 10 epoch آموزش دادیم و نتایج آن در اشکال 2-14 و 2-15 قابل مشاهده است. در این نمودار ها مشخص است که مدل بعد از 3 ایپاک بر روی دادگان آموزش کاملا fit شده ولی دقت آن بر روی داده های آموزش نشان میدهد که این مدل با اینکه در حداکثر ظرفیت خود است، عملکرد ضعیفتری نسبت به مدل قبلی دارد.



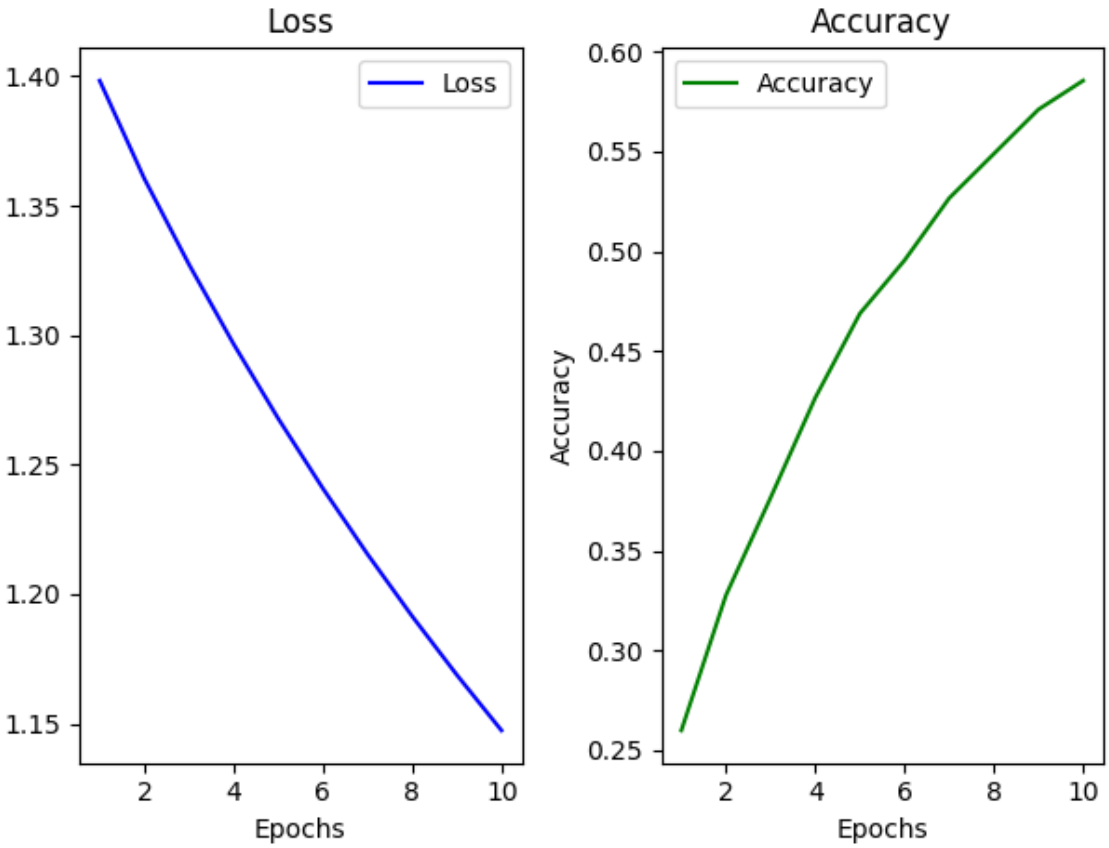
شکل 2-14. نمودار آموزش با آزاد کردن همه لایه ها

	precision	recall	f1-score	support
NEUTROPHIL	0.92	0.88	0.90	25
EOSINOPHIL	0.85	0.88	0.86	25
MONOCYTE	1.00	0.96	0.98	25
LYMPHOCYTE	0.96	1.00	0.98	25
accuracy			0.93	100
macro avg	0.93	0.93	0.93	100
weighted avg	0.93	0.93	0.93	100

شکل 2-15. نمودار آزمون با آزاد کردن همه لایه ها

2-4-2. لایه دسته بند

با آزاد کردن دسته بند، تنها 4100 پارامتر از کل پارامتر ها (معادل 0.06 درصد) آزادند. دقت این مرحله به وضوح از مراحل قبل کمتر است ولی نمودار خطا و دقت نشان میدهد که مدل میتواند با آموزش بیشتر به نتایج بهتری برسد. به علت این که در مقاله و در مراحل قبل از 10 مرحله برای آموزش استفاده شده بود، نتایج را در همین تعداد epoch نگه داشتیم تا بتوان مقایسه کرد، ولی میبینیم که سرعت همگرایی این مرحله بسیار کمتر است. یک دلیل مهم آن است که بر خلاف ViT، این مدل توانایی کار با 4 لایه نداشت و مجبور شدیم لایه جدیدی را دستی اضافه کنیم، این باعث شد که به جای finetune، دسته بند در وضعیت آموزش قرار بگیرد و سرعت همگرایی کمی داشته باشد. نتایج در اشکال 2-16 و 2-17 قابل مشاهده هستند.



شکل 2-9. نمودار آموزش با آزاد کردن لایه دسته بند

	precision	recall	f1-score	support
NEUTROPHIL	0.56	0.60	0.58	25
EOSINOPHIL	0.40	0.40	0.40	25
MONOCYTE	0.55	0.68	0.61	25
LYMPHOCYTE	0.71	0.48	0.57	25
accuracy			0.54	100
macro avg	0.55	0.54	0.54	100
weighted avg	0.55	0.54	0.54	100

شکل 2-10. نمودار آزمون با آزاد کردن لایه دسته بند

2-5. تحلیل و نتیجه گیری

با تحلیل نتایج که در جدول 2-1 آورده شده، در می یابیم که مدل های ViT با 2 لایه آخر و ViT کامل بیشترین دقت را داشتند. این در حالتی است که CNN با آموزش کامل تنها از مدل ViT که فقط دسته بند آزاد داشت بهتر عملکرده و متوجه میشویم که عملکرد ViT بهتر از CNN است.

در شرایط دادگان که نویز نسبتا زیادی دارند و ابعاد تصاویرشان کوچک است، ترنفورمر ها بهتر از مدل های کانولوشنی عمل میکنند زیرا معماری ترنفورمر ها به گونه ای است که تصاویر را به قسمت های کوچکتر میشکند و با استفاده از attention، کل تصویر را به صورت یکپارچه و کلی میبینند، درنتیجه تاثیر نویز را کمتر میکند. همچنین CNN ها عکس را یک بار در هر لایه میبینند اما ترنسفورمر ها چند بار یک تصویر را میبینند (feature reuse) و میتوانند در لایه های مختلف abstraction یک تصویر را تحلیل کنند، در نتیجه robust تر هستند. لازم به ذکر است که پیچیدگی ترنسفورمر ها بیشتر است و منابع بیشتری لازم دارند.

Method	*PERCISION*	*RECALL*	*F1-SCORE*
ViT-Classifier	0.76	0.76	0.76
ViT-First 2 Layers	0.94	0.94	0.94
ViT-Last 2 Layers	0.97	0.97	0.97
ViT-Full	0.97	0.97	0.97
CNN-Full	0.93	0.93	0.93
CNN-Classifier	0.55	0.54	0.54

جدول 2-1. جمع‌بندی نتایج پرسش 2