|  |  |  |
| --- | --- | --- |
| **پرسش ۱** | **نام و نام خانوادگی** | علی صفری |
| **شماره دانشجویی** | 810202153 |
| **پرسش ۲** | **نام و نام خانوادگی** | حمیدرضا نادی مقدم |
| **شماره دانشجویی** | ۸۱۰۱۰۳۲۶۴ |
|  | **مهلت ارسال پاسخ** | **۱۴۰3.10.13** |

|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس شبکه‌های عصبی و یادگیری عمیق**  **تمرین پنجم** | | |

**فهرست**

[**مقدمه** 1](#_Toc187010004)

[**پرسش 1.** پیشبینی نیروی باد به کمک مبدل و تابع خطایHuber 2](#_Toc187010005)

[۱-۱. مقدمه 2](#_Toc187010006)

[2-1. آماده‌سازی 4](#_Toc187010007)

[1-2-1. Autoencoder 4](#_Toc187010008)

[2-2-1. مکانیزم توجه و Positional encoding 9](#_Toc187010009)

[3-2-1. تابع Huber 11](#_Toc187010010)

[4-2-1. الگوریتم Slime mould 13](#_Toc187010011)

[3-1. روش شناسی و نتایج 18](#_Toc187010012)

[1-3-1. انتخاب دیتا و نمایش شکل 6 مقاله 18](#_Toc187010013)

[2-3-1. شناسایی داده‌های پرت 19](#_Toc187010014)

[3-3-1. دینویز و پیش پردازش داده‌ها (5 + 15 نمره) 21](#_Toc187010015)

[4-3-1. پیاده‌سازی مدل با 3 معماری و 2 تابع loss برای حالت single step 27](#_Toc187010016)

[5-3-1. بررسی و ارزیابی عملکرد مدل‌ها 32](#_Toc187010017)

[6-3-1. مقادیر بهینه ابرپارامتر با استفاده از slime mould (امتیازی) 39](#_Toc187010018)

[7-3-1. پیاده سازی مدل مبدل با استفاده از دو تابع loss برای حالت multi step 43](#_Toc187010019)

[**پرسش۲. استفاده از ViT برای طبقهبندی تصاویر گلبولهای سفید** 47](#_Toc187010020)

[1-2. آماده‌سازی داده‌ها 47](#_Toc187010021)

[2-2. آموزش مدل‌ها 51](#_Toc187010022)

[3-۲. تحلیل و نتیجه گیری 59](#_Toc187010023)

**شکل‌ها**

شکل 1- تصویر شماتیک مقاله ازencoder و decoder 6

شکل 2 - کد ساخت Autoencoder بخش اول 8

شکل 3- کد ساخت Autoencoder بخش دوم 9

شکل 4 - فرمول Huber loss 11

شکل 5- پیاده‌سازی تابع Huber loss 12

شکل 6 - تصویر الگوریتم SMA از مقاله 15

شکل 7 - پیاده‌سازی SMA بخش اول 16

شکل 8- پیاده‌سازی SMA بخش دوم 17

شکل 9 - کد مربوط به پیاده سازی شکل 6 مقاله 18

شکل 10- نمایش شکل 6 مقاله 19

شکل 11 - کد نوشته شده برای تشخیص داده‌های پرت 20

شکل 12 - نمودار تشخیص داده‌های پرت (مشابه شکل 7 مقاله) 21

شکل 13 - کد مربوط به دینوز کردن داده‌ 25

شکل 14- نمودار داده‌های دینویز شده و داده‌های اولیه (مشابه شکل 8 مقاله) 26

شکل 15- خروجی داده آموزش و تست بعد از نرمال کردن و sliding window برای single step 27

شکل 16- کد ساخت مدل transformer 28

شکل 17 - کد ساخت مدل RNN 29

شکل 18- کد ساخت مدل MLP 30

شکل 19- نمونه خروجی فرایند آموزش 32

شکل 20- جدول مقایسه پارامترهای بدست آمده برای هر مدل مشابه جدول 3 مقاله 34

شکل 21 - جدول مقایسه پارامترهای بدست آمده برای هر مدل مشابه جدول 3 مقاله (one feature) 37

شکل 22 – کد تعریف کلاس بهینه‌سازی هایپرپارامترها 41

شکل 23 - نتایج بهنیه کردن هایپرپارامترهای مدل مبدل 42

شکل 24- کد ساخت مدل transformer برای multi step 44

شکل 25 - آماده‌سازی مدل برای آموزش mutli step 44

شکل 26 - بخشی از فرایند آموزش مدل mutli step 45

شکل 27 - نتایج مدل transformer در حالت mutli step 46

شکل 28- کد نمایش تصادفی یک تصویر از هر کلاس 47

شکل 29- نمایش تصادفی یک تصویر از هر کلاس 48

شکل 30- تعداد تصاویر در هر کلاس 48

شکل 31- متوازن کردن کلاس ها با تقویت داده 50

شکل 32- تقسیم داده به دو قسمت آموزش و ارزیابی 51

شکل 33- بارگذاری مدل ViT و نمایش معماری آن 52

شکل 34- فقط دسته¬بند Classifier قابل آموزش باشد 54

شکل 35- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت یک 55

شکل 36- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت دو 55

شکل 37- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت سه 56

شکل 38- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت چهار 56

شکل 39- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت پنج 57

شکل 40- نمودار Accuracy و Loss برای مدل CCN آموزش فقط در لایه طبقه بند 58

شکل 41- مقایسه مدل CNN و ViT با آموزش در لایه طبقه بند 58

**جدول‌ها**

[جدول 1- مقایسه مقاله و نتایج ما در single step 35](#_Toc187010065)

[جدول 2- مقایسه مقاله و نتایج ما در single step (one feature) 38](#_Toc187010066)

# **مقدمه**

برای پیاده‌سازی پروژه از بستر Google Colab به منظور کد نویسی و اجرا استفاده شده است. تمامی مراحل کد و اجرای آن در این گزارش به تفصیل شرح داده شده است.

کد های نوشته شده همگی در پوشه‌ی Code و با پسوند .ipynd ذخیره شده است.

# **پرسش 1.** پیشبینی نیروی باد به کمک مبدل و تابع خطایHuber

## **۱-۱. مقدمه**

**محدودیت‌های روش‌های آماری سنتی مانند GARCH و ARIMA**

1. **فرضیات سختگیرانه درباره توزیع داده‌ها**: روش‌های آماری مانند GARCH و ARIMA بر پایه پیش‌فرض‌های توزیع مشخص و هموار بودن داده‌ها طراحی شده‌اند، که در مواردی که داده‌ها ناپایدار یا دارای نویز باشند، کارایی این روش‌ها کاهش می‌یابد.
2. **نیاز به آزمایش‌های هموارسازی(smoothness test) داده‌ها**: این روش‌ها برای تحلیل صحیح نیازمند انجام آزمایش‌هایی مانند هموارسازی و بررسی ایستایی داده‌ها هستند، که این فرآیند زمان‌بر است و مدل را به شرایط خاص محدود می‌کند.
3. **محدودیت در تعمیم‌پذیری**: این مدل‌ها در شناسایی وابستگی‌های بلندمدت و پیچیدگی‌های غیرخطی ناکارآمد هستند، و قابلیت تطبیق با داده‌های متنوع و محیط‌های متغیر را ندارند.
4. **کاهش دقت در شرایط ناپایدار**: تغییرات شدید و نوسانات در داده‌ها می‌توانند به کاهش عملکرد این مدل‌ها منجر شوند.

**مزایای مدل‌های یادگیری ماشین مانند Random Forest، SVM و XGBoost**

1. **انعطاف‌پذیری در مواجهه با داده‌های متنوع**: این مدل‌ها می‌توانند ویژگی‌های مختلف و داده‌های بزرگ و پیچیده را مدیریت کنند.
2. **دقت بالاتر پیش‌بینی**: توانایی یادگیری الگوهای پیچیده باعث بهبود دقت پیش‌بینی نسبت به روش‌های سنتی می‌شود.
3. **عدم نیاز به پیش‌فرض‌های توزیعی سختگیرانه**: این مدل‌ها در مقایسه با روش‌های آماری محدودیت کمتری در پیش‌فرض‌های داده دارند.
4. **عملکرد بهتر در داده‌های نویزدار**: این مدل‌ها توانایی مدیریت نویزها و داده‌های پرت را دارند.

**چگونگی رفع محدودیت‌های روش‌های یادگیری ماشین سنتی توسط مدل‌های یادگیری عمیق مانند GRU، LSTM و CNN**

1. **مدیریت وابستگی‌های بلندمدت**: مدل‌هایی مانند GRU و LSTM قادر به شناسایی و یادگیری وابستگی‌های بلندمدت در داده‌های سری زمانی هستند.
2. **استخراج ویژگی‌های پیچیده**: مدل CNN می‌تواند الگوهای محلی و ویژگی‌های مهم داده‌های پیچیده را با استفاده از فیلترهای کانولوشنی شناسایی کند.
3. **کارایی بهتر در مسائل غیرخطی**: این مدل‌ها برای داده‌های غیرخطی و پیش‌بینی‌هایی که به تحلیل دقیق‌تر وابستگی‌های زمانی نیاز دارند، عملکرد بسیار بهتری نسبت به مدل‌های سنتی ارائه می‌دهند.
4. **پیش‌بینی دقیق‌تر در داده‌های سری زمانی**: این مدل‌ها می‌توانند با دقت بیشتری ویژگی‌های زمانی را تحلیل کنند.

**اهمیت مکانیزم خودتوجهی (Self-Attention) در شبکه‌های مبدل (Transformer)**

1. **مدل‌سازی وابستگی‌های بلندمدت**: این مکانیزم وابستگی‌های طولانی در داده‌های سری زمانی را شناسایی و مدل‌سازی می‌کند.
2. **استخراج اطلاعات در مقیاس‌های مختلف زمانی**: مکانیزم خودتوجهی همبستگی‌های محلی (local) و جهانی (global) داده‌ها را استخراج می‌کند.
3. **بهبود کارایی در داده‌های پیچیده**: این مکانیزم در ترکیب با دیگر مدل‌ها عملکرد پیش‌بینی را به‌ویژه در سری‌های زمانی پیچیده بهبود می‌بخشد.
4. **کاهش وابستگی به توالی داده‌های مجاور**: برخلاف RNN، این مکانیزم اطلاعات کل توالی را بدون توجه به موقعیت مکانی پردازش می‌کند**.**

**چگونگی افزایش پایداری و استحکام مدل‌های پیش‌بینی داده‌های باد فراساحلی توسط تابع خطای هیوبر (Huber)**

1. **مدیریت داده‌های پرت و نوسانات شدید**: تابع خطای Huber از ترکیب ویژگی‌های تابع خطای مربعات و قدرمطلق بهره می‌برد و حساسیت کمتری به داده‌های پرت دارد، که در داده‌های پرنوسان باد فراساحلی رایج است.
2. **افزایش پایداری فرآیند بهینه‌سازی**: این تابع پیوستگی و مشتق‌پذیری دارد، که فرآیند تنظیم پارامترهای مدل را با روش‌هایی مانند گرادیان نزولی پایدارتر و کارآمدتر می‌کند.
3. **توازن بین دقت و استحکام**: استفاده از این تابع خطا باعث کاهش اثرات منفی نویزها و داده‌های پرت شده و پیش‌بینی‌های دقیق‌تر و پایدارتر را برای داده‌های متغیر باد فراساحلی فراهم می‌آورد.
4. **افزایش دقت پیش‌بینی**: با کاهش تاثیر نویز و داده‌های پرت، دقت پیش‌بینی به‌ویژه در داده‌های پرنوسان بهبود می‌یابد.

## **2-1. آماده‌سازی**

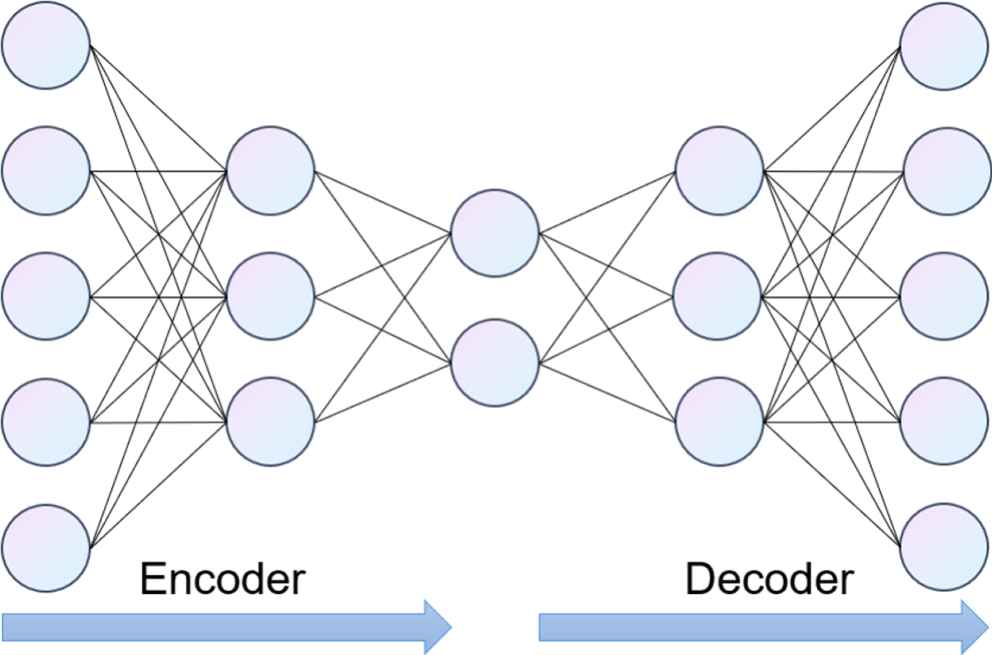
### **1-2-1. Autoencoder**

**هدف استفاده از ساختار Autoencoder**

* هدف اصلی استفاده از Autoencoder در این تحقیق، بازسازی و کاهش نویز داده‌های باد فراساحلی است.
* این ساختار برای شناسایی ویژگی‌های مهم داده‌ها و کاهش اثر نویز و داده‌های پرت طراحی شده است.
* به کمک این ساختار، داده‌ها بازسازی شده و بهبود دقت پیش‌بینی مدل‌های یادگیری عمیق ممکن می‌شود.

**بخش‌های تشکیل‌دهنده Autoencoder**

* **بخش Encoder** داده‌های ورودی را به یک فضای با ابعاد کمتر (نمایش مخفی) تبدیل می‌کند.
* **بخش Decoder** داده‌های کدگذاری شده را بازسازی می‌کند تا به داده‌های اصلی نزدیک شوند.
* **تابع هزینه (Loss Function)** میزان تفاوت بین داده‌های اصلی و بازسازی‌شده را اندازه‌گیری می‌کند.



شکل 1- تصویر شماتیک مقاله ازencoder و decoder

برای پیاده‌سازی کد این بخش از کتابخانه tensorflow استفاده کردیم. در این پروژه، معماری خودرمزنگار (Autoencoder) به صورت زیر طراحی شده است:

**Encoder**

* تعداد ابعاد ورودی یا نورون‌های لایه ورودی با توجه به اندازه پنجره (window size) که در اینجا ۱۴۴ است، تعریف شده است.
* داده‌ها در گام اول با استفاده از لایه Flatten به صورت یک بردار مسطح درمی‌آیند.
* یک لایه مخفی (hidden layer) با ۱۲۸ نورون و تابع فعال‌سازی ReLU معرفی شده است.
* در گام بعد، یک لایه مخفی دیگر با ۶۴ نورون و تابع فعال‌سازی ReLU به شبکه اضافه شده است.
* سپس لایه Latent تعریف شده که به عنوان فضای فشرده‌شده با تعداد نورون‌های تعیین‌شده در اینجا latent\_dim و تابع فعال‌سازی ReLU استفاده شده است.

**Decoder**

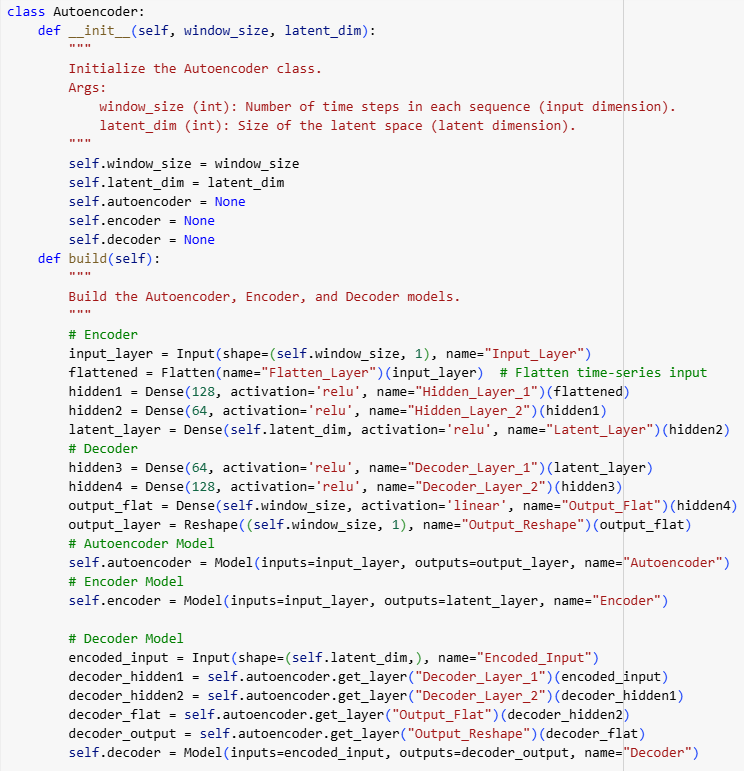
* پس از کدگذاری داده‌ها در لایه Latent، مراحل بازسازی (Decoding) آغاز می‌شود.
* لایه‌ای با ۶۴ نورون و تابع فعال‌سازی ReLU برای بازسازی داده‌ها به کار گرفته شده است.
* یک لایه دیگر با ۱۲۸ نورون و همان تابع فعال‌سازی اضافه شده است.
* در پایان، یک لایه خروجی (output layer) تعریف شده که با استفاده از Reshape، داده‌ها را به شکل اصلی سری زمانی (۱۴۴ گام زمانی) بازمی‌گرداند.

**ساختار نهایی:**

* داده‌ها پس از عبور از لایه‌های Encoder فشرده‌شده و نویزهای ناخواسته از بین می‌روند.
* سپس، داده‌ها در Decoder بازسازی شده و سری زمانی دینویزشده تولید می‌شود.

**مزیت این معماری:** این معماری با بهره‌گیری از لایه‌های متعدد و کاهش تدریجی تعداد نورون‌ها، امکان یادگیری ویژگی‌های مهم سری زمانی را فراهم کرده و با استفاده از لایه Latent، داده‌ها را به صورت فشرده

درآورده و نویز را حذف می‌کند. همچنین، بازسازی داده‌ها در Decoder تضمین می‌کند که سری زمانی خروجی با حداقل افت اطلاعات بازسازی شود.



شکل 2 - کد ساخت Autoencoder بخش اول

A screenshot of a computer program

Description automatically generated

شکل 3- کد ساخت Autoencoder بخش دوم

### 2-2-1. مکانیزم توجه و Positional encoding

**مکانیزم توجه (Attention Mechanism) در شبکه مبدل**

مکانیزم توجه یکی از اجزای کلیدی شبکه مبدل (Transformer) است و برای **شناسایی و مدل‌سازی وابستگی‌های مهم بین قسمت‌های مختلف داده‌های سری زمانی** استفاده می‌شود.

**هدف استفاده از مکانیزم توجه:**

1. **شناسایی روابط مهم**: مکانیزم توجه اطلاعاتی را که در بخش‌های مختلف داده مهم هستند شناسایی می‌کند و به آن‌ها وزن بیشتری اختصاص می‌دهد.
2. **مدل‌سازی وابستگی‌های بلندمدت**: برخلاف مدل‌های سنتی مانند RNN که فقط وابستگی به نقاط نزدیک را در نظر می‌گیرند، مکانیزم توجه می‌تواند تمام نقاط در یک توالی را به صورت همزمان پردازش کند و وابستگی‌های بلندمدت را کشف کند.
3. **افزایش دقت پیش‌بینی**: مکانیزم توجه اجازه می‌دهد که مدل بخش‌های مهم داده را شناسایی کرده و از آن‌ها برای پیش‌بینی دقیق‌تر استفاده کند.
4. **کاهش هزینه محاسباتی**: در مقایسه با مدل‌های بازگشتی (RNN)، این مکانیزم امکان پردازش موازی داده‌ها را فراهم می‌کند که محاسبات را سریع‌تر می‌کند.

**Positional Encoding در شبکه مبدل**

شبکه مبدل مکانیزم توجه را به کار می‌گیرد، اما این مکانیزم به تنهایی ترتیب داده‌ها را در نظر نمی‌گیرد. Positional Encoding به مدل کمک می‌کند تا **موقعیت هر عنصر در توالی را تشخیص دهد**.

**هدف استفاده از Positional Encoding**

1. **حفظ اطلاعات ترتیب داده‌ها**: از آنجا که مکانیزم توجه به صورت ذاتی اطلاعات ترتیب (ordering) را ندارد، Positional Encoding تضمین می‌کند که مدل بتواند موقعیت نسبی هر عنصر را درک کند.
2. **مدل‌سازی وابستگی‌های زمانی**: این مکانیزم کمک می‌کند تا مدل روابط بین عناصر با فاصله زمانی مختلف را بهتر درک کند.
3. **بهبود کارایی مدل در داده‌های سری زمانی** :Positional Encoding اطلاعات اضافی در مورد موقعیت نسبی داده‌ها را به مدل اضافه می‌کند، که در داده‌های سری زمانی مانند پیش‌بینی باد فراساحلی بسیار مفید است.
4. **حفظ ارتباط فضایی در داده‌های ورودی**: این تکنیک تضمین می‌کند که مدل بتواند اطلاعات ساختاری مربوط به توالی را به صورت کامل حفظ کند.

### 3-2-1. تابع Huber

تابع خطای **Huber** ترکیبی از **خطای مربعات میانگین** (Mean Squared Error) و **خطای قدرمطلق** (Mean Absolute Error) است. این تابع به صورت زیر عمل می‌کند:

* اگر خطا کوچک باشد (کمتر از مقدار آستانه δ)، رفتار تابع مانند **MSE** است.
  + در این حالت، تابع حساسیت بیشتری به خطاهای کوچک دارد و به تنظیم دقیق مدل کمک می‌کند.
* اگر خطا بزرگ باشد بیشتر از مقدار آستانه δ، رفتار تابع مانند **MAE** است.
  + در این حالت، تابع از تأثیر زیاد داده‌های پرت جلوگیری می‌کند و مدل را پایدارتر می‌سازد.

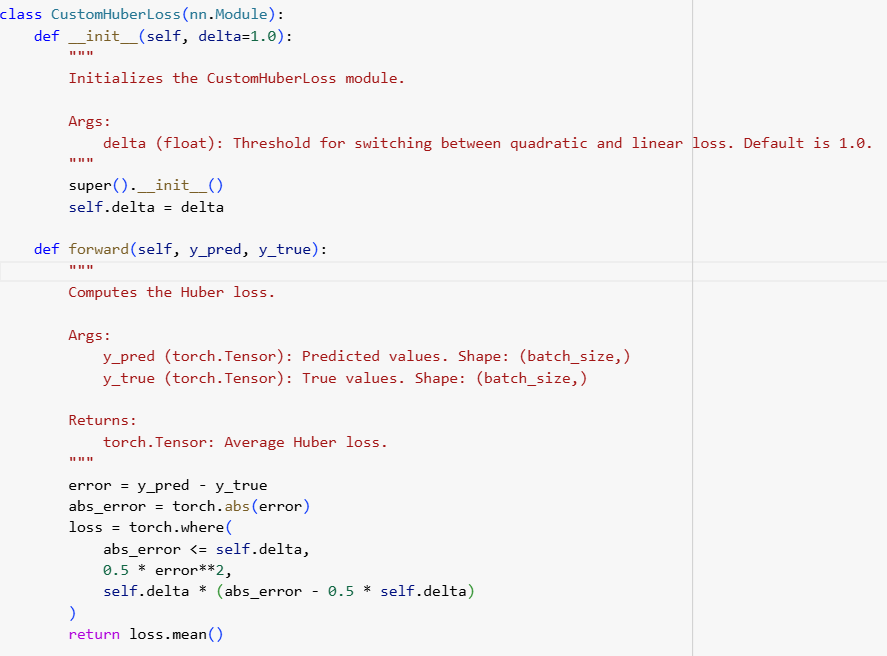
A white background with black text

Description automatically generated

شکل 4 - فرمول Huber loss

**هدف استفاده از تابع خطای Huber**

1. **مدیریت داده‌های پرت**: این تابع تأثیر داده‌های پرت را محدود می‌کند، که در داده‌های ناپایدار مانند باد فراساحلی بسیار مفید است.
2. **توازن بین دقت و پایداری**: با ترکیب مزایای MSE و MAE، این تابع هم دقت بالایی ارائه می‌دهد و هم از مشکلات ناشی از خطاهای بزرگ جلوگیری می‌کند.
3. **پایداری بهینه‌سازی**: مشتق‌پذیری تابع باعث می‌شود که فرآیند بهینه‌سازی با روش‌هایی مانند گرادیان نزولی پایدارتر انجام شود.



شکل 5- پیاده‌سازی تابع Huber loss

کلاس CustomHuberLoss در چارچوب PyTorch طراحی شده و برای محاسبه (Huber Loss) به کار می‌رود. در ادامه توضیحاتی درباره ورودی و خروجی این تابع ارائه شده است:

**ورودی‌ها:**

1. **y\_pred پیش‌بینی مدل:**
   * یک Tensor است که مقادیر پیش‌بینی‌شده توسط مدل را شامل می‌شود.
   * شکل (Shape) آن برابر با (batch\_size,) است، به این معنا که هر مقدار در این آرایه مربوط به یک نمونه از داده‌ها در دسته (Batch) است.
2. **y\_true مقادیر واقعی:**
   * یک Tensor که مقادیر هدف یا مقادیر واقعی مربوط به داده‌ها را در خود دارد.
   * شکل آن نیز مشابه (batch\_size,) است و باید با y\_pred هم‌اندازه باشد.

**خروجی‌ها:**

1. **Huber Loss** 
   * خروجی تابع یک اسکالر (Scalar) است که مقدار میانگین Huber loss را برای تمام نمونه‌های موجود در Batch محاسبه می‌کند.
   * Huber loss ترکیبی از loss خطی و درجه دوم است که بستگی به مقدار خطا (Error) دارد. اگر خطا کمتر از مقدار آستانه (delta) باشد، زیان به صورت درجه دوم محاسبه می‌شود و در غیر این صورت، خطی خواهد بود.

**نکات کلیدی:**

* مقدار delta تعیین می‌کند که چه زمانی تابع از حالت درجه دوم به حالت خطی تغییر کند.
* این تابع به‌ویژه در مواردی که داده‌های پرت (Outliers) وجود دارند، عملکرد بهتری نسبت به زیان مربع خطا (MSE) دارد زیرا تأثیر داده‌های پرت را کاهش می‌دهد.
* ورودی‌های y\_pred و y\_true باید از نوع torch.Tensor باشند و بر روی یک دستگاه (Device) قرار داشته باشند CPU یا GPU

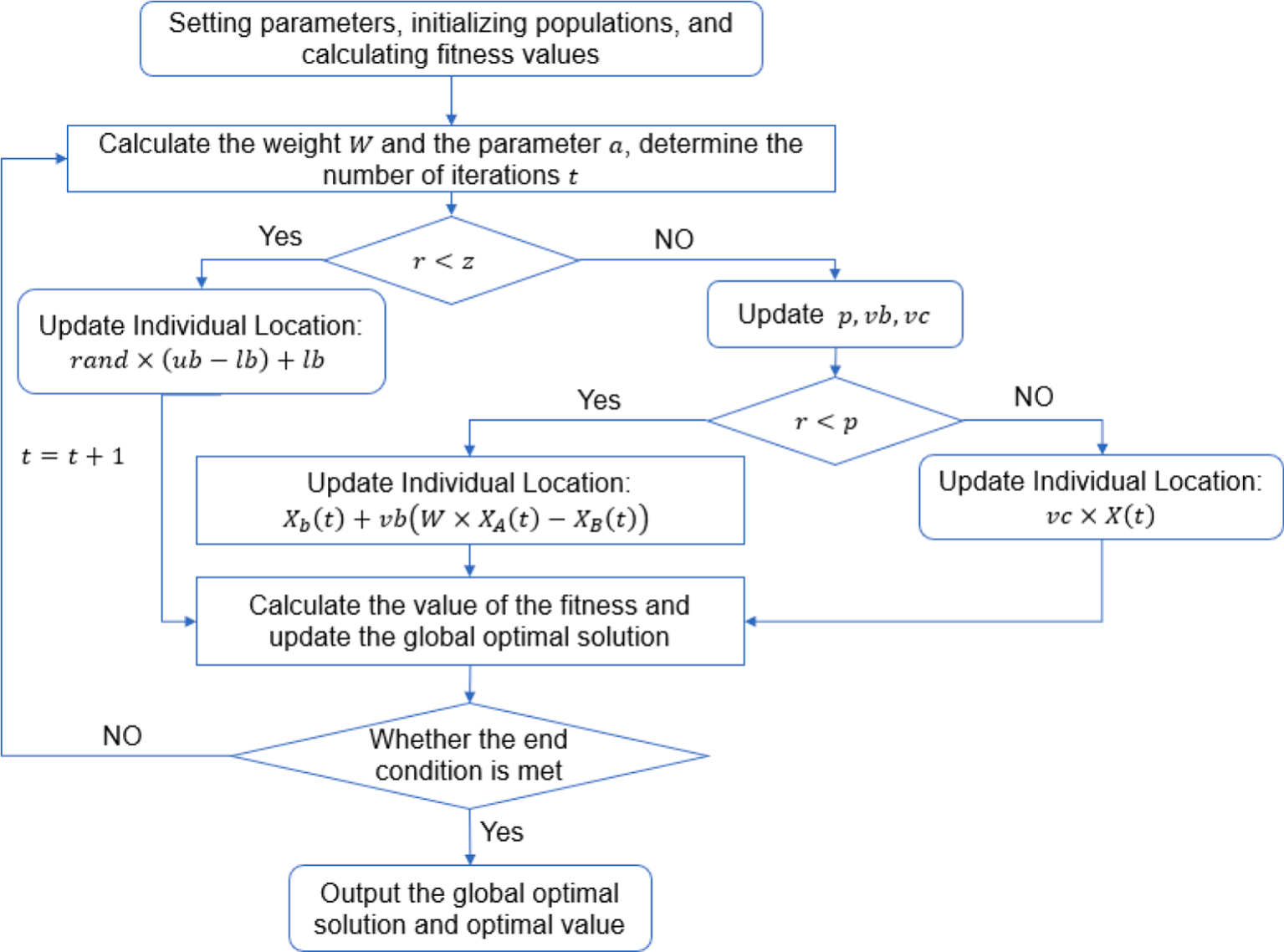
این تابع به عنوان یک ماژول PyTorch زیرکلاس nn.Module قابل استفاده است و به راحتی می‌توان آن را در مدل‌های یادگیری عمیق به کار برد.

### 4-2-1. الگوریتم Slime mould

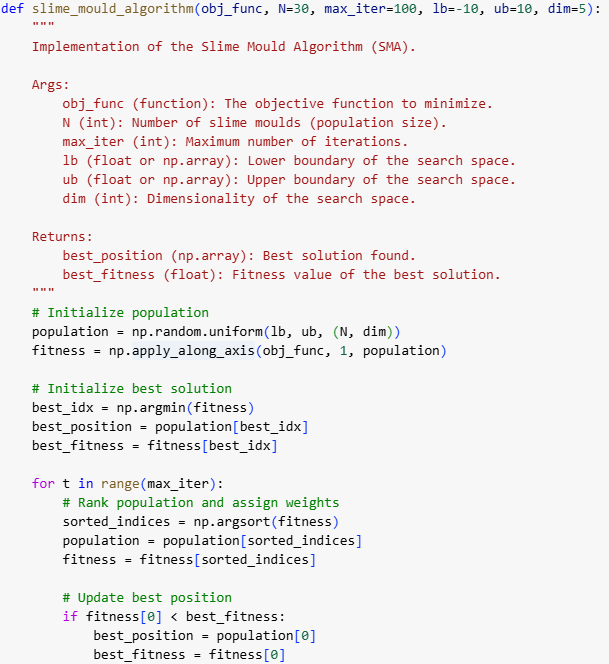
**Slime Mould Algorithm (SMA)** یک الگوریتم الهام‌گرفته از طبیعت است که از رفتار لجن کپک در جستجوی غذا تقلید می‌کند. این الگوریتم برای حل مسائل بهینه‌سازی طراحی شده است و با شبیه‌سازی رفتارهای کاوش و بهره‌برداری لجن کپک، می‌تواند به راه‌حل‌های بهینه دست یابد.

**مراحل الگوریتم:**

1. **مقدمه‌سازی اولیه**:
   * جمعیتی از موقعیت‌های تصادفی در فضای جستجو تولید می‌شود.
   * محدوده‌های جستجو lb و ubو پارامترهای مسئله تعریف می‌شوند.
2. **ارزیابی تناسب**:
   * مقدار تناسب هر موقعیت (بر اساس تابع هدف) محاسبه می‌شود.
3. **محاسبه وزن‌ها**:
   * لجن کپک‌ها بر اساس رتبه‌بندی تناسب خود وزن دریافت می‌کنند.
4. **به‌روزرسانی موقعیت‌ها**:
   * موقعیت‌ها در دو فاز بهره‌برداری و کاوش به‌روزرسانی می‌شوند:
     + **بهره‌برداری (Exploitation)** حرکت به سمت بهترین موقعیت (بیشترین منابع غذا).
     + **کاوش (Exploration)** حرکت تصادفی برای جلوگیری از گیر افتادن در بهینه محلی.
5. **تکرار فرآیند**:
   * این فرآیند تا رسیدن به حداکثر تعداد تکرار یا تحقق معیار توقف ادامه می‌یابد.



شکل 6 - تصویر الگوریتم SMA از مقاله



شکل 7 - پیاده‌سازی SMA بخش اول

A screenshot of a computer program

Description automatically generated

شکل 8- پیاده‌سازی SMA بخش دوم

**توضیحات پارامترها (آرگومان‌های ورودی):**

1. **obj\_func:** تابع هدف که باید بهینه شود. در این مثال از تابع کره‌ای استفاده شده است.
2. **N** : اندازه جمعیت (تعداد slime mould)
3. **max\_iter** : حداکثر تعداد تکرارهای الگوریتم.
4. **lb** و **ub** : محدوده پایین و بالا برای متغیرهای مسئله.
5. **dim** : تعداد ابعاد یا متغیرهای مسئله.

این کد با استفاده از رفتارهای طبیعی slime mould و با توجه به الگوریتم نوشته شده در مقاله طراحی شده و می‌تواند برای مسائل مختلف بهینه‌سازی استفاده شود. البته لازم به ذکر است این تابع برای کاربرد عمومی نوشته شده است و در بخش امتیازی بسته به مسئله تغییر اندکی خواهد داشت که در همان بخش اشاره خواهد شد.

لازم به ذکر است که کد مربوط به این بخش و بخش امتیازی در فایلی تحت عنوان HW5\_1\_Slime ذخیره شده است.

## **3-1. روش شناسی و نتایج**

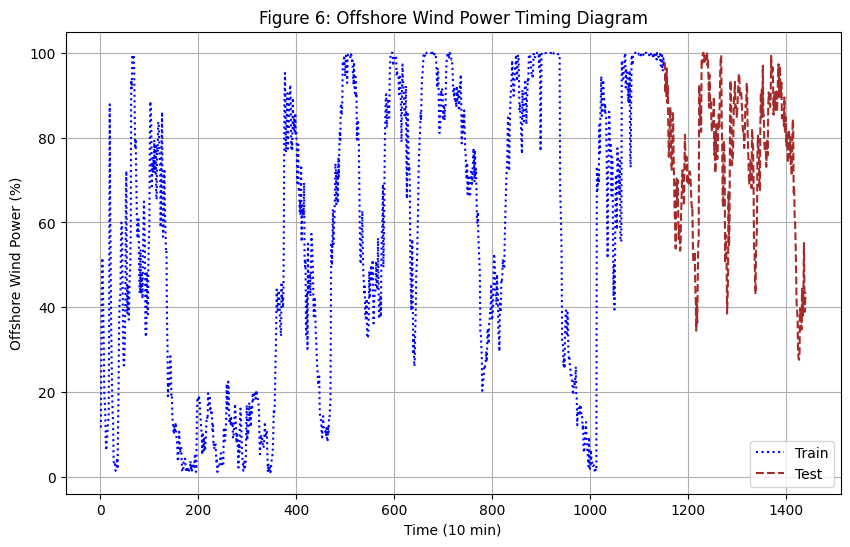
### 1-3-1. انتخاب دیتا و نمایش شکل 6 مقاله

برای این کار از لینک قرار داده شده در مقاله دیتای مورد استفاده که در فایل WT5 بود را دانلود کرده و 1440 دیتای اول آن را استفاده کردیم. این فایل در فولدر اصلی ضمیمه شده است. در این قسمت 80 درصد ابتدایی به عنوان داده آموزش و 20 درصد انتهایی به عنوان داده تست در نظر گرفته شده اند.

A screenshot of a computer code

Description automatically generated

شکل 9 - کد مربوط به پیاده سازی شکل 6 مقاله



شکل 10- نمایش شکل 6 مقاله

### 2-3-1. شناسایی داده‌های پرت

کدی که نوشته‌ام برای شناسایی داده‌های پرت (Outliers) در مجموعه داده‌های توان بادی دریایی طراحی شده است. این الگوریتم بر اساس روش پنجره متحرک (Rolling Window) کار می‌کند. در این روش، پنجره‌ای با اندازه 48 نقطه در نظر گرفته شده که شامل 24 نقطه قبل و 24 نقطه بعد از نقطه مورد بررسی است. برای هر پنجره، میانگین (μ) و انحراف معیار (σ) داده‌های موجود محاسبه می‌شود و داده‌هایی که از بازه μ ± 2σ فراتر باشند، به عنوان داده پرت شناسایی و علامت‌گذاری می‌شوند.

در مقایسه با روش ارائه‌شده در مقاله، که از واریانس (σ²) به جای انحراف معیار استفاده کرده بود، تغییراتی انجام داده‌ام. استفاده از انحراف معیار به دلیل سازگاری بیشتر با مقیاس داده‌ها و سهولت تفسیر آن، انتخاب شده است. همچنین، پنجره به صورت متقارن تنظیم شده است تا هر دو سمت نقطه داده به طور مساوی بررسی شود. این تنظیم به شناسایی دقیق‌تر الگوهای محلی و رفتار داده‌ها کمک می‌کند. هرچند این جزئیات در مقاله به وضوح مشخص نشده بود، اما این تغییرات برای تحلیل داده‌های سری زمانی همچون این مجموعه منطقی و ضروری است.

برای نمایش نتایج، از نمودار پراکندگی (Scatter Plot) استفاده کرده‌ام و نقاط پرت با رنگ زرد مشخص شده‌اند. این روش بصری‌سازی داده‌ها را بهبود داده و تحلیل نتایج را آسان‌تر می‌کند. در مجموع، با وجود شباهت به روش مقاله، تغییرات اعمال‌شده باعث ساده‌تر شدن فرایند تحلیل و تطابق بهتر الگوریتم با ماهیت این داده‌ها شده است.

A screenshot of a computer program

Description automatically generated

شکل 11 - کد نوشته شده برای تشخیص داده‌های پرت

A graph showing a wind power timing

Description automatically generated with medium confidence

شکل 12 - نمودار تشخیص داده‌های پرت (مشابه شکل 7 مقاله)

### 3-3-1. دینویز و پیش پردازش داده‌ها (5 + 15 نمره)

گزارش جامع فرایند دینویز و پیش‌پردازش داده‌های توان بادی فراساحلی

در این بخش، داده‌های توان بادی فراساحلی پس از گردآوری اولیه، با هدف بهبود کیفیت و آماده‌سازی برای پیش‌بینی چندمرحله‌ای، تحت مراحل زیر قرار می‌گیرند:

* دینویز داده‌ها با شبکه خودرمزنگار (Autoencoder)
* تقسیم داده‌ها به مجموعه‌های آموزش و آزمون
* نرمال‌سازی داده‌ها
* تبدیل داده‌ها به متغیرهای ویژگی و پاسخ با پنجره متحرک

انتخاب و رعایت ترتیب مناسب این مراحل نقش مهمی در حفظ تعمیم‌پذیری مدل و جلوگیری از نشت داده‌ها ایفا می‌کند. در ادامه، به توضیح گام‌های فوق و مفروضات انجام‌شده می‌پردازیم.

**حذف نویز با شبکه خودرمزنگار**

در این مرحله، داده‌های خام با استفاده از یک شبکه خودرمزنگار بازسازی شده و نویزهای ناخواسته آن‌ها حذف می‌گردد. در این پروژه، برای ورودی خودرمزنگار از پنجره‌های ۱۴۴ گام زمانی استفاده شده است؛ بدین معنا که برای هر بخش ۱۴۴ تایی از داده، مدل تلاش می‌کند خروجی آن را بازسازی (دینویز) نماید. البته لازم به ذکر است اینکه پنچره ورودی چه ابعادی در بخش دینویز کردن داشته باشد در مقاله ذکر نشده بود اما به جهت اینکه در آموزش مدل 144 استفاده شده در این بخش هم از همین عدد استفاده گردیده است.

**معماری خودرمزنگار:**

* **لایه ورودی** : (144, 1)
* **Flatten:**  برای کاهش ابعاد داده‌ها
* **لایه‌های پنهان:** با 128 و 64 نورون
* **لایه کدگذاری (Latent Layer)** : با ابعاد مشخص‌شده (Latent Dim)
* **لایه‌های دیکدینگ:** 64 و 128 نورون
* **خروجی بازسازی‌شده:** با استفاده از Reshape به شکل اصلی داده‌ها

**فرض:** استفاده از پنجره ۱۴۴ گامی برای حذف نویز و بازسازی داده‌ها به‌منظور دستیابی به اطلاعات بلندمدت‌تر و بهبود پیش‌بینی‌های چندمرحله‌ای.

**تقسیم داده به مجموعه‌های آموزش و آزمون**

پس از اتمام مرحله دینویز، داده‌های دینویزشده (به طول کل ۱۴۴۰) به دو بخش آموزش و آزمون تقسیم می‌شوند:

* **۸۰٪ داده‌ها:** (بخش ابتدایی سری زمانی) برای آموزش
* **۲۰٪ داده‌ها:** (بخش انتهایی) برای آزمون

این ترتیب تضمین می‌کند که مدل در مرحله آموزش هیچ دسترسی مستقیم یا غیرمستقیم به داده‌های آزمون نداشته باشد و از نشت اطلاعات جلوگیری شود.

**نرمال‌سازی داده‌ها**

برای هم‌مقیاس‌کردن متغیرها و تسهیل فرایند آموزش شبکه‌های عصبی، داده‌های آموزش و آزمون به‌صورت جداگانه نرمال‌سازی می‌شوند. در این پروژه، از روش MinMaxScaler با بازه ‌[0, 1] استفاده شده است:

* جدا کردن داده‌های آموزش و آزمون
* فیت‌کردن MinMaxScaler فقط روی داده‌های آموزش
* اعمال همان اسکالر روی داده‌های آزمون

لازم به ذکر است که در انتها و برای ارزیابی به صورت معکوس نرمال‌سازی انجام شده است تا به ساختار اولیه بازگردیم.

**فرض:** انتخاب MinMaxScaler برای داده‌هایی که در دامنه محدودی قرار دارند و از توزیع ناهمگون برخوردارند، مناسب است.

**ایجاد متغیرهای ویژگی و پاسخ با پنجره ۱۴۴ گامی**

پس از نرمال‌سازی، داده‌های دینویزشده و مقیاس‌شده برای پیش‌بینی چندمرحله‌ای به‌صورت زیر پردازش می‌شوند:

* **پنجره ویژگی (X)** ۱۴۴ گام زمانی متوالی که حاوی 8 فیچر است.
* **افق پیش‌بینی (Horizon)** : در مقاله ۲۴ گام زمانی آینده مطرح شده است اما در سوال ما دو حالت multi step و single step داریم. که در ادامه توضیح داده خواهد شد
* **خروجی (Y)** سری زمانی مورد پیش‌بینی در این گام مورد نظر (در صورتی که single step باشد در یک گام بعدی و در شرایطی که multi باشد تا جایی که مورد نظر سوال است.)

**فرایند:**

* داده‌های آموزش:
  + قطعه‌بندی در پنجره‌های متوالی ۱۴۴ تایی برای تشکیل ورودی (X)
  + گام یا گام‌های آینده برای تشکیل خروجی (Y)
* داده‌های آزمون:
  + اعمال همان سازوکار برای ساخت X\_test و Y\_test

**فرض:** این مقادیر انتخابی هستند و بسته به ماهیت داده‌ها و اهداف پروژه می‌توانند تغییر کنند.

**اهمیت ترتیب مراحل**

رعایت ترتیب زیر در پیش‌پردازش طبق توصیه‌های علمی ضروری است:

1. **دینویز داده‌ها:** ابتدا حذف نویز برای بهبود کیفیت داده‌ها
2. **تقسیم به آموزش و آزمون:** جلوگیری از نشت اطلاعات آزمون
3. **نرمال‌سازی:** اسکالر فقط روی داده‌های آموزش فیت می‌شود
4. **تشکیل متغیرهای X و Y با پنجره متحرک**

**توجه:** عدم رعایت ترتیب ممکن است باعث نشت اطلاعات داده‌های آزمون به مرحله آموزش شود.

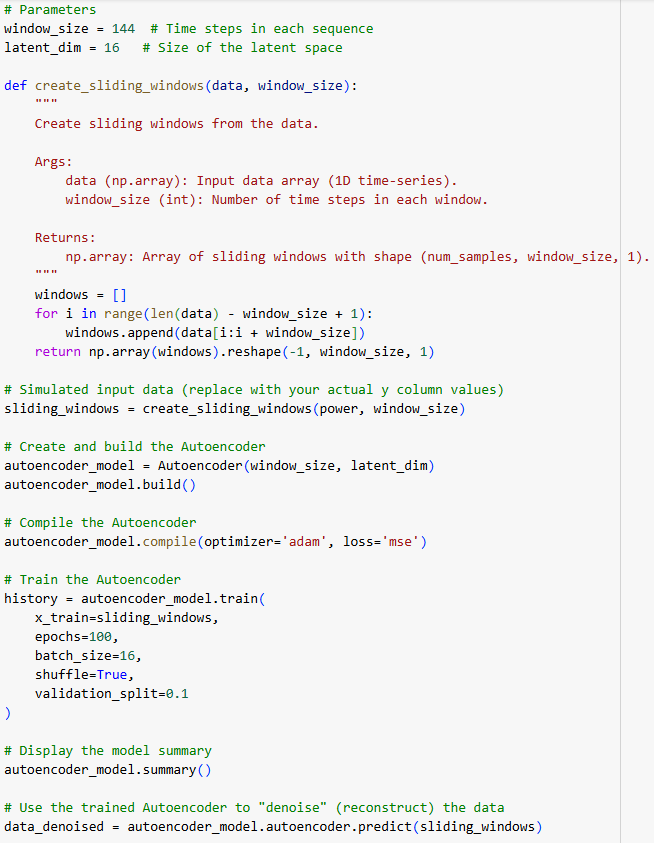
**مثال نقض**

برای توضیح اهمیت ترتیب مراحل، فرض کنید ابتدا داده‌ها نرمال‌سازی شوند و سپس به مجموعه‌های آموزش و آزمون تقسیم گردند. در این حالت، اسکالر بر روی کل داده‌ها (شامل داده‌های آزمون) فیت می‌شود. این موضوع باعث می‌شود که مدل به اطلاعات آماری داده‌های آزمون دسترسی پیدا کند. نتیجه این امر، کاهش واقعی چالش پیش‌بینی و افزایش مصنوعی دقت مدل است. اما در زمان مواجهه با داده‌های جدید، مدل نمی‌تواند عملکرد مشابهی ارائه دهد و دقت آن به شدت کاهش می‌یابد.

**جمع‌بندی**

* داده‌های توان بادی با استفاده از شبکه خودرمزنگار و پنجره ۱۴۴ گامی نویززدایی شدند.
* داده‌های دینویزشده به دو بخش آموزش (۸۰٪) و آزمون (۲۰٪) تقسیم شدند.
* نرمال‌سازی داده‌ها با MinMaxScaler انجام شد.
* با استفاده از پنجره متحرک ۱۴۴ گامی و افق پیش‌بینی به تعداد گام مورد سوال ، متغیرهای ویژگی (X) و پاسخ (Y) ایجاد شدند.

این ترتیب تضمین‌کننده آموزش بدون نشت داده آزمون و دستیابی به پیش‌بینی‌های دقیق و تعمیم‌پذیر برای توان بادی است.



شکل 13 - کد مربوط به دینوز کردن داده‌

**گزارش درباره تجمیع و هم‌پوشانی در Sliding Window**

در تحلیل داده‌های ترتیبی با Sliding Window، داده‌های بازسازی‌شده با هم‌پوشانی پنجره‌ها تجمیع می‌شوند. برای هر پنجره، مقادیر به موقعیت متناظر در آرایه بازسازی‌شده اضافه و شمارشگر برای هر موقعیت بروزرسانی می‌شود:

denoised\_full[i:i + window\_size] += data\_denoised[i].flatten()

count[i:i + window\_size] += 1

در پایان، مقادیر هر موقعیت با تعداد مشارکت‌ها تقسیم شده و میانگین‌گیری انجام می‌شود:

denoised\_full /= count

این فرآیند نویز را کاهش داده و داده بازسازی‌شده‌ای با کیفیت بالا تولید می‌کند.

**مدیریت نویز و کاربردها**

پیش از آموزش، این روش با هم‌پوشانی داده‌ها دقت پیش‌بینی را افزایش می‌دهد. پس از آموزش، داده‌های بازسازی‌شده بدون نویز اضافی ارائه می‌شوند. این روش با تنظیم اندازه پنجره و گام حرکتی به تعادل دقت و کارایی کمک می‌کند و کیفیت داده‌های ورودی و خروجی را بهبود می‌بخشد.

A graph showing a graph

Description automatically generated with medium confidence

شکل 14- نمودار داده‌های دینویز شده و داده‌های اولیه (مشابه شکل 8 مقاله)

A white background with black text

Description automatically generated

شکل 15- خروجی داده آموزش و تست بعد از نرمال کردن و sliding window برای single step

### 4-3-1. پیاده‌سازی مدل با 3 معماری و 2 تابع loss برای حالت single step

**مدل Transformer**

مدل Transformer با استفاده از معماری پیشرفته خود برای داده‌های ترتیبی طراحی شده است. جزئیات معماری به شرح زیر است:

* **Positional Encoding** : استفاده از کدگذاری موقعیتی برای افزودن اطلاعات ترتیبی به داده‌ها.
* **ابعاد Embedding** : 64
* **تعداد هدهای توجه (Attention Heads)** : 4
* **تعداد لایه‌های Encoder: 3**
* **ابعاد شبکه Feedforward** : 128
* **حداکثر طول توالی (max\_len)** : 144
* **لایه خروجی (Output Layer)** : یک لایه خطی برای پیش‌بینی نهایی.
* **تابع فعال‌سازی:** ReLU برای شبکه‌های میانی.

این تنظیمات تضمین می‌کند که مدل بتواند الگوهای پیچیده زمانی را شناسایی کند.

A screenshot of a computer program

Description automatically generated

شکل 16- کد ساخت مدل transformer

مدل RNN مدل RNN طراحی شده برای شناسایی وابستگی‌های ترتیبی در داده‌ها، با مشخصات زیر است:

* **تعداد لایه‌ها:**  3
* **تعداد واحدهای مخفی (Hidden Units)**: 128
* **نوع RNN**: RNN پایه با tanh به عنوان تابع فعال‌سازی.
* **ورودی**: داده‌ها با شکل (batch\_size, window\_size, input\_dim) وارد مدل شده و پس از پردازش، خروجی به شکل (batch\_size, hidden\_dim) تولید می‌شود.
* **لایه‌های Fully Connected**: شامل یک لایه میانی با ابعاد 64 واحد مخفی که تابع فعال‌سازی ReLU بر روی آن اعمال شده و یک لایه خروجی خطی برای پیش‌بینی نهایی.

به‌علاوه، در مدل RNN جدید، با افزایش تعداد واحدهای مخفی و اضافه کردن یک لایه Fully Connected میانی، مدل توانایی یادگیری الگوهای پیچیده‌تری را به دست آورده است.

A screenshot of a computer program

Description automatically generated

شکل 17 - کد ساخت مدل RNN

**مدل MLP**

مدل MLP برای پردازش داده‌های ورودی مسطح طراحی شده و معماری آن به صورت زیر است:

* **لایه‌های مخفی:** دو لایه با اندازه‌های 64 و 32 واحد.
* **تابع فعال‌سازی (Activation Function):** ReLUس
* **لایه خروجی:** یک لایه خطی برای تولید پیش‌بینی نهایی.



شکل 18- کد ساخت مدل MLP

**توابع loss**

* **MSE:** برای محاسبه میانگین مربع اختلاف بین مقادیر پیش‌بینی‌شده و واقعی.
* **Huber Loss:** برای کاهش تاثیر داده‌های پرت با استفاده از ترکیبی از MSE و MAE.

**تنظیمات آموزش**

**داده‌ها و پیش‌پردازش:**

* داده‌ها به دو مجموعه آموزشی و آزمایشی تقسیم شده و با استفاده از scaler نرمال‌سازی می‌شوند.
* **تعداد ویژگی‌های ورودی:** 8 ویژگی شامل:
* Sequence No.
* V
* D
* Air Density
* Humidity
* I
* S\_a
* S\_b
* استفاده از train\_loader برای مدیریت داده‌های ورودی به مدل‌ها.

**پارامترهای آموزش:**

* **تعداد دوره‌ها (Epochs)** : 80 (ابتدا 50 مشابه مقاله در نظر گرفتیم اما در نهایت دیدم با افزایش دوره عملکرد مدل بهبود میابد به همین جهت افزایش دادیم.)
* **نرخ یادگیری (Learning Rate)** : 0.001
* **بهینه‌ساز:** Adam برای تنظیم وزن‌ها در تمام مدل‌ها.

**حلقه آموزش:**

1. **مقدمه مدل:**
   * هر ترکیب مدل و loss با نمونه جدیدی از مدل شروع می‌شود.
2. **(Forward Pass)**:
   * داده‌های ورودی از مدل عبور کرده و پیش‌بینی‌ها محاسبه می‌شوند.
3. **محاسبه loss:** 
   * اختلاف بین مقادیر پیش‌بینی‌شده و واقعی با استفاده از تابع loss انتخابی.
4. **(Backward Pass):** 
   * گرادیان‌ها محاسبه شده و وزن‌ها با استفاده از بهینه‌ساز Adam به‌روزرسانی می‌شوند.
5. **ثبت دوره‌ها:**
   * ثبت loss برای هر دوره و نمایش نتایج در فواصل منظم.

A screenshot of a computer

Description automatically generated

شکل 19- نمونه خروجی فرایند آموزش

### 5-3-1. بررسی و ارزیابی عملکرد مدل‌ها

در این قسمت، برنامه‌ای برای ارزیابی و مقایسه کارایی انواع مدل‌های توسعه یافته ارایه شده است. در این راستا، سه نوع مدل Transformer، RNN و MLP مورد استفاده قرار گرفتند. هر یک از این مدل‌ها با استفاده از دو نوع تابع MSE و Huber آزمون شده و متریک‌های RMSE، MAE و MAPE محاسبه شدند.

**روش فعالیت:**

برای پیاده‌سازی و اجرای این برنامه، کارهای زیر انجام شد:

1. **تعریف و استخراج متریک‌ها:** مقادیر واقعی و پیش‌بینی‌شده با استفاده از RMSE، MAE و MAPE ارزیابی شدند. برای اطمینان از دقت، در محاسبه به موانع مثل تقسیم بر صفر توجه شد.
2. **تعریف مدل‌ها:** در بخش قبل مفصل توضیح داده شد.
3. **روش آموزش:**
   * آموزش هر مدل به مدت 80 دوره با استفاده از optimizer نوع Adam و سرعت یادگیری 0.001 انجام شد.
   * مقادیر خطا هر 10 دوره گزارش شد.
4. **ارزیابی نهایی:**
   * از داده‌های آزمون برای مقایسه مقادیر واقعی و پیش‌بینی‌شده استفاده شد. خروجی‌ها به مقیاس اولیه بازگردانده شدند.
   * متریک‌ها روی مقیاس اولیه محاسبه شدند.

**نتایج نهایی:**

جداول به دو قالب Wide و Long(فرمت مقاله) ارایه شدند. قالب Wide خلاصه‌ای از نتایج هر مدل با استفاده از تابع ازدستدهی مختلف ارایه می‌دهد. قالب Long به مقایسه‌ی جزئی‌تر بین متریک‌ها می‌پردازد.

A screenshot of a computer

Description automatically generated

شکل20- جدول مقایسه پارامترهای بدست آمده برای هر مدل مشابه جدول 3 مقاله

در ادامه به صورت اضافه مقایسه ای از نتایج بدست آمده و نتایج مقاله خواهیم داشت.س

جدول 1- مقایسه مقاله و نتایج ما در single step

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Huber | MSE | Huber | | MSE | Model | index |
| مقاله | | | نتایج ما | |  |  |
| 3.1067 | 3.2837 | 2.6196 | | 9.6248 | MLP | MAE |
| 2.9013 | 2.9225 | 4.8347 | | 2.88 | RNN |  |
| 2.2485 | 2.3670 | 8.8134 | | 5.5385 | Transformer |  |
| 4.22 | 4.82 | 6.4974 | | 24.0203 | MLP | MAPE |
| 4.22 | 4.25 | 11.8551 | | 6.9850 | RNN |  |
| 3.22 | 3.63 | 21.5450 | | 13.3821 | Transformer |  |
| 4.0115 | 4.3644 | 3.6604 | | 10.2629 | MLP | RMSE |
| 3.5891 | 3.6151 | 5.5852 | | 3.5387 | RNN |  |
| 2.7764 | 2.8817 | 9.8435 | | 7.1588 | Transformer |  |

در مقایسه نتایج مدل‌ها، MLP تحت Huber عملکرد خوبی در معیارهای MAE و RMSE داشت و حتی در RMSE توانست نتایجی نزدیک به مقاله ارائه دهد. مدل RNN نیز تحت MSE در معیارهای MAE و RMSE به نتایج مقاله نزدیک بود و نشان داد که می‌تواند به‌خوبی الگوهای موجود در داده‌ها را یاد بگیرد. اگرچه در برخی معیارها، به‌ویژه MAPE، عملکرد ضعیف‌تری نسبت به مقاله داشت، اما توانایی کلی آن در پیش‌بینی قابل توجه است. مدل Transformer با وجود تفاوت در نتایج، توانست در معیارهای MAE و RMSE تحت MSE به‌خوبی عمل کند و عملکرد نسبی قابل توجهی نشان داد. این نتایج نشان می‌دهد که مدل‌ها در یادگیری الگوهای زمانی موفق عمل کرده‌اند و امکان بهبود عملکرد آن‌ها با تنظیمات بهتر وجود دارد.

**بخش اضافی**

با توجه به عملکرد ‌نچندان عالی مدل در بخش قبل من تصمیم گرفتم تنها از y به عنوان فیچر ورودی استفاده کنم و عملکرد مدل را در حالتی که تنها به تاریخچه قدرت باد نگاه می‌کنیم، بررسی کنم. در این حالت ورودی ما تنها یک فیچر خواهد داشت و به جای ابعاد 144 × 8 ابعاد 144 × 1 خواهیم داشت. در تحلیل من، این نتیجه‌گیری که استفاده از تنها خروجی y به عنوان ورودی x می‌تواند نتایج بهتری نسبت به رویکرد مقاله که از ۸ ویژگی مختلف مانند "V"، "D"، "air density"، "Humidity"، و غیره استفاده کرده، داشته باشد، نشان‌دهنده اهمیت عمیق‌تر فهم روابط بین داده‌ها است.

دلایل احتمالی برای عملکرد بهتر مدل من می‌تواند شامل موارد زیر باشد:

۱. **سادگی داده‌های ورودی**: با استفاده از تنها y، مدل من توانسته است به طور کامل روی الگوهای زمانی y تمرکز کند و از ورود نویز یا اطلاعات غیرمرتبط که ممکن است توسط سایر ویژگی‌ها ایجاد شود، جلوگیری کند. این سادگی به مدل اجازه داده که بهتر روی روابط مستقیم و غیرخطی بین زمان‌های گذشته و آینده تمرکز کند.

۲. **کیفیت پایین ویژگی‌های اضافی**: ممکن است ویژگی‌هایی که در مقاله استفاده شده‌اند، ارتباط قوی و مستقیمی با مقدار yنداشته باشند. در صورتی که این ویژگی‌ها نویزی یا کم‌کیفیت باشند، مدل قادر به استخراج الگوهای معنادار از آن‌ها نخواهد بود و در نتیجه عملکرد کاهش می‌یابد.

۳. **کاهش پیچیدگی و جلوگیری از بیش‌برازش**: استفاده از ورودی ساده‌تر یعنی فقط y می‌تواند باعث شود مدل، بیش‌برازش روی داده‌های آموزشی را کاهش دهد و در نتیجه عملکرد بهتری روی داده‌های تست داشته باشد. این موضوع به‌ویژه زمانی صادق است که تعداد نمونه‌های آموزشی کم باشد.

۴. **وابستگی‌های زمانی قوی در y:** ممکن است y به تنهایی الگوهای زمانی کافی برای پیش‌بینی دقیق گام‌های بعدی را در خود داشته باشد. به این ترتیب، مدل نیازی به اطلاعات اضافی از ویژگی‌های دیگر ندارد.

با این حال، باید به نکات زیر نیز توجه کرد:

* **ریسک overfit شدن بهy:** اگر مدل تنها به y وابسته باشد، ممکن است عملکرد خوبی روی این دیتاست خاص داشته باشد، اما در سناریوهای دنیای واقعی یا داده‌های دیگر که اطلاعات محیطی بیشتری لازم است، عملکرد افت کند.
* **نادیده گرفتن اطلاعات ارزشمند**: ممکن است برخی ویژگی‌ها، مانند "air density" یا "humidity"، اطلاعات ارزشمندی را در بازه‌های زمانی مختلف ارائه دهند. حذف این اطلاعات می‌تواند در سناریوهای پیچیده‌تر محدودیت‌هایی ایجاد کند.

در نهایت، من معتقدم که نتایج بهتر مدل من نشان می‌دهد که داشتن تعداد زیادی ویژگی همیشه منجر به بهبود عملکرد نمی‌شود. در واقع، درک درست از روابط بین داده‌ها، کیفیت پیش‌پردازش و مهندسی ویژگی‌ها، و استفاده بهینه از مدل اهمیت بیشتری دارد. برای تایید این نتایج، انجام آزمایش‌های بیشتر روی دیتاست‌های متفاوت و بررسی تأثیر ویژگی‌های حذف‌شده ضروری به نظر می‌رسد. این تحلیل می‌تواند به درک بهتر نقش ویژگی‌های اضافی در پیش‌بینی توان بادی کمک کند. لازم به ذکر است کد این بخش از پیاده سازی هم در پوشه Code تحت عنوان HW5\_1\_ one\_feature ذخیره شده است. در ادامه نتایج بدست آمده در این روش گزارش شده است.

A table of numbers and symbols

Description automatically generated

شکل 21 - جدول مقایسه پارامترهای بدست آمده برای هر مدل مشابه جدول 3 مقاله (one feature)

جدول 2- مقایسه مقاله و نتایج ما در single step (one feature)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Huber | MSE | Huber | | MSE | Model | index |
| مقاله | | | نتایج ما | |  |  |
| 3.1067 | 3.2837 | 1.3557 | | 1.1864 | MLP | MAE |
| 2.9013 | 2.9225 | 0.4282 | | 0.6414 | RNN |  |
| 2.2485 | 2.3670 | 1.0332 | | 1.4603 | Transformer |  |
| 4.22 | 4.82 | 2.1155 | | 1.7475 | MLP | MAPE |
| 4.22 | 0.0425 | 0.6600 | | 0.9428 | RNN |  |
| 3.22 | 3.63 | 1.4817 | | 2.0537 | Transformer |  |
| 4.0115 | 4.3644 | 1.6344 | | 1.5324 | MLP | RMSE |
| 3.5891 | 3.6151 | 0.4803 | | 0.7070 | RNN |  |
| 2.7764 | 2.8817 | 1.1785 | | 1.5402 | Transformer |  |

در این مقایسه، عملکرد سه مدل **MLP**،**RNN** و **Transformer** بر اساس معیارهای **MAE**، **MAPE** و **RMSE** بررسی شده است. نتایج نشان می‌دهد پیاده‌سازی ما در تمام معیارها عملکرد بهتری نسبت به مقاله داشته است.

**MAE** : در مدل MLP، مقدار خطا از 3.1067 (مقاله) به 1.3557 (Huber) کاهش یافته است. مدل‌های RNN و Transformer نیز به ترتیب کاهش‌های قابل‌توجهی حدود 85% و 50% در MAE نشان داده‌اند.

**MAPE** : در مقاله مقادیر به‌صورت خام (غیر درصدی) گزارش شده‌اند. پس از تبدیل به درصد، MAPE برای مدل MLP از 4.22% به 2.1155% کاهش یافت. در RNN و Transformer نیز مقادیر به‌ترتیب از 4.22% و 3.22% به 0.6600% و 1.4817% کاهش یافته‌اند.

**RMSE** : برای MLP، خطا از 4.0115 (مقاله) به 1.6344 کاهش یافته است. مدل‌های RNN و Transformer نیز بیش از 80% و 50% بهبود در این معیار نشان داده‌اند.

نتایج نشان‌دهنده بهبود قابل‌توجه پیاده‌سازی ما در تمامی مدل‌ها و معیارها بوده و برتری روش‌های بهینه‌سازی ما را تایید می‌کند. عملکرد بهتر مدل احتمالاً به دلیل **پیش‌پردازش بهتر داده‌ها** مثل استفاده از autoencoder برای کاهش نویز، **تنظیم دقیق‌تر معماری مدل‌ها** با انتخاب hyperparameterهای مناسب است.

### 6-3-1. مقادیر بهینه ابرپارامتر با استفاده از slime mould (امتیازی)

محتوای بخش امتیازی

در این پروژه، هدف استفاده از **مدل Transformer** برای پیش‌بینی سری‌های زمانی و بهینه‌سازی ابرپارامترهای آن با استفاده از **الگوریتم Slime Mould Algorithm (SMA)** است. مدل Transformer یکی از پیشرفته‌ترین مدل‌های یادگیری عمیق برای پردازش داده‌های توالی‌محور است که در این پروژه برای پیش‌بینی چندگامی به‌کار گرفته شده است.

**معماری مدل Transformer**

مدل Transformer شامل بخش‌های کلیدی زیر است:

* **Positional Encoding**: برای اضافه کردن اطلاعات زمانی به داده‌های ورودی
* **Multi-head Attention**: برای یادگیری وابستگی‌های پیچیده در داده‌ها
* **FeedForward Network**: برای پردازش ویژگی‌های استخراج‌شده  
  این مدل به‌طور خاص برای پیش‌بینی چندمرحله‌ای طراحی شده است و می‌تواند چندین مقدار خروجی را به‌صورت هم‌زمان پیش‌بینی کند.

**ابرپارامترهای بهینه‌شده**

ابرپارامترهای کلیدی که در این پروژه بهینه‌سازی شده‌اند، عبارتند از:

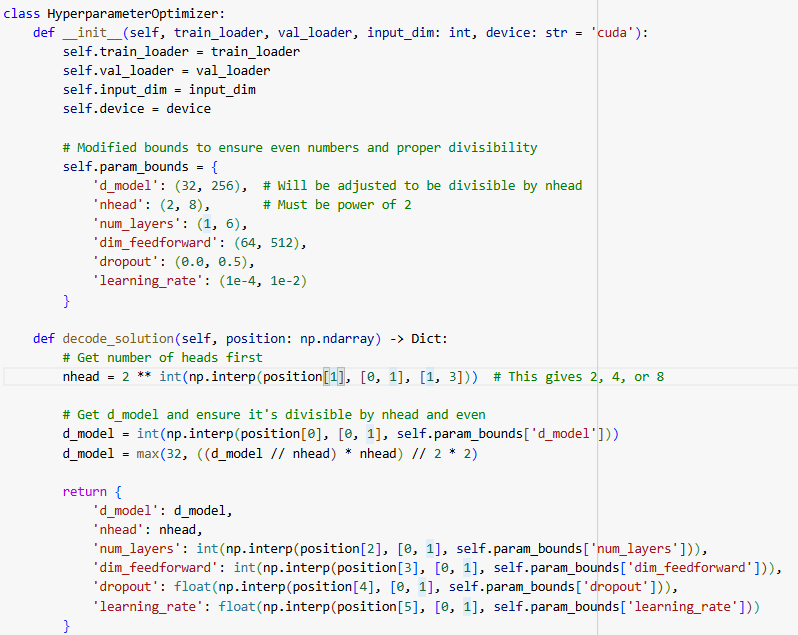
* **Nhead**: تعداد سرهای توجه
* **d\_model**: ابعاد مدل
* **num\_layers**: تعداد لایه‌ها
* **dim\_feedforward**:اندازه لایه FeedForward
* **dropout**: نرخ ریزش
* **learning rate**: نرخ یادگیری

**الگوریتم Slime Mould Algorithm (SMA)**

الگوریتم SMA یک روش فراابتکاری الهام‌گرفته از رفتار کپک مخاطی است که برای بهینه‌سازی مسائل پیچیده استفاده می‌شود. در این پروژه، از این الگوریتم برای تنظیم مقادیر بهینه ابرپارامترهای مدل Transformer استفاده شده است.

**مراحل اجرای SMA**

1. **ایجاد جمعیت اولیه**: مقادیر اولیه ابرپارامترها به‌صورت تصادفی تولید می‌شوند.
2. **آموزش مدل**: مدل Transformer با هر ترکیب از ابرپارامترها آموزش داده می‌شود.
3. **ارزیابی عملکرد**: خطای اعتبارسنجی (Validation Loss) برای هر ترکیب محاسبه می‌شود.
4. **به‌روزرسانی جمعیت**:
   * ترکیب‌ها براساس خطای اعتبارسنجی مرتب می‌شوند.
   * اعضای جمعیت براساس منطق الگوریتم SMA به سمت بهترین ترکیب حرکت می‌کنند.
   * مقادیر خارج از محدوده به مقادیر مجاز بازگردانده می‌شوند.
5. **تکرار مراحل تا همگرایی**: فرآیند تا زمانی که به بهترین مقادیر برسیم، تکرار می‌شود.



شکل 20 – کد تعریف کلاس بهینه‌سازی هایپرپارامترها

**ویژگی‌های کد**

* **تنظیم خودکار ابرپارامترها**: الگوریتم SMA بهترین ترکیب ابرپارامترها را با کمترین خطای اعتبارسنجی ارائه می‌دهد.
* **توقف زودهنگام (Early Stopping)**: اگر خطای اعتبارسنجی بهبود نیابد، آموزش زودتر متوقف می‌شود.
* **سازگاری مدل**: اطمینان حاصل می‌شود که d\_modeld\\_modeld\_model همواره بر nheadnheadnhead بخش‌پذیر باشد.

**نتایج**

1. بهترین مقادیر برای ابرپارامترها، از جمله:
   * **d\_model**
   * **nhead**
   * **num\_layers**
   * **dim\_feedforward**
   * **dropout**
   * **learning rate**  
     به همراه کمترین خطای اعتبارسنجی گزارش می‌شود.
2. زمان کل بهینه‌سازی محاسبه شده و کارایی الگوریتم مشخص می‌شود.
3. بهینه‌سازی نشان داده که استفاده از الگوریتم‌های فراابتکاری مانند SMA برای تنظیم ابرپارامترهای مدل‌های پیچیده‌ای مانند Transformer می‌تواند منجر به دستیابی به عملکرد بهینه شود.

استفاده از الگوریتم‌های فراابتکاری مانند SMA می‌تواند راه‌حلی کارآمد برای تنظیم ابرپارامترهای مدل‌های پیچیده باشد. مدل Transformer با ابرپارامترهای بهینه، توانست پیش‌بینی‌های چندمرحله‌ای با دقت بالا انجام دهد و الگوریتم SMA نقش مهمی در یافتن مقادیر بهینه داشت.

در نهایت با پیاده‌سازی نتیجه به فرم زیر در آمده و هایپرپارامترهای بهینه مدل اینگونه گزارش می‌شوند.

A screenshot of a computer

Description automatically generated

شکل 21 - نتایج بهنیه کردن هایپرپارامترهای مدل مبدل

### 7-3-1. پیاده سازی مدل مبدل با استفاده از دو تابع loss برای حالت multi step

برای پیاده‌سازی این حالت معماری کلی مدل تغییری نمی‌کند (غیر از خروجی) بلکه چگونگی بکار گیری مدل و ارسال داده به مدل متفاوت می‌شود. تا پیش از این و برای حالت single step ما یک پنجره 144 تایی از ورودی‌ها را به همراه y گام بعد به شبکه می‌دادیم و تا آموزش برای همان یک گام بعد از 144 گام صورت گیرد و در بخش ارزیابی هم همین روند را پیش بردیم. اما در این بخش با تغییر پارامتر forcast\_step یا forecast\_horizon به جای عدد یک، یک لیست از اعداد شام 4و 8 و 16 را خواهیم داشت و این بدان معناست که در آموزش مدل 144 گام داده می‌شود و برای y مقادیر 4 گام بعد، 8 گام بعد و 16 گام بعد هم به مدل داده می‌شود تا الگوی این فرایند را آموزش ببیند. بدین صورت مدل توانایی پیشبینی در این گام‌ها را خواهد داشت.

در این رویکرد:

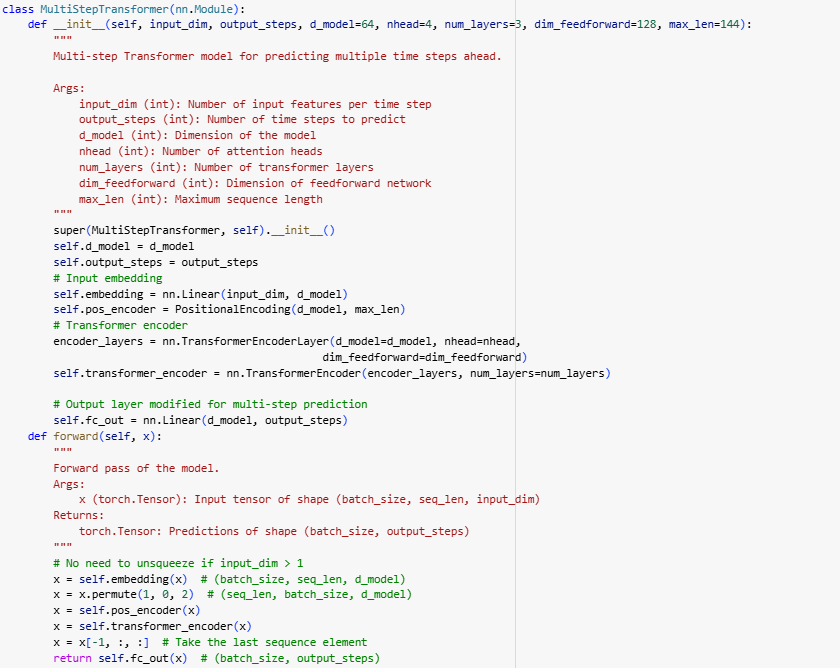
مدل پیش‌بینی‌ها را برای تمام افق‌های مشخص‌شده در یک مرحله محاسبه (forward pass) تولید می‌کند مثلاً یک بردار با اندازه 3 برای t+4، t+8، و t+16

برخلاف روش‌های پیش‌بینی تکراری، مدل پیش‌بینی‌های خود را برای پیش‌بینی مراحل بعدی به ورودی بازنمی‌گرداند، که این موضوع از انباشت خطا جلوگیری می‌کند.

هدف آموزشی تضمین می‌کند که مدل الگوها را در تمام افق‌های زمانی به صورت هم‌زمان یاد بگیرد و توانایی خود را برای پیش‌بینی چندمرحله‌ای بهینه کند.

این ساختار به مدل این امکان را می‌دهد که مستقیماً مقادیر مربوط به مراحل زمانی آینده را بر اساس پنجره 144 مرحله‌ای ورودی پیش‌بینی کند.

* همچنین با توجه به اینکه هر گام در دیتاست معادل 10 دقیقه است هر کدام از این گام های گفته شده **t+4**، **t+8** و **t+16** به ترتیب معادل **40 دقیقه**، **80 دقیقه** و **160 دقیقه** بعد هستند. و در حالت تک گام تنها در حال پیشبینی 10 دقیقه بعد بودیم.



شکل 22- کد ساخت مدل transformer برای multi step

A screenshot of a computer program

Description automatically generated

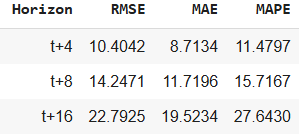
شکل 23 - آماده‌سازی مدل برای آموزش mutli step

A screenshot of a computer

Description automatically generated

شکل 24 - بخشی از فرایند آموزش مدل mutli step

در نهایت مدل را بعد از آموزش با داده تست ارزیابی کرده و جدولی مشابه جدول 5 مقاله ایجاد کردیم. البته چون در این بخش سوال الزامی به ایجاد جدول دقیقا مشابه مقاله و در کد نشده بود صرفا جدولی که داده‌ها را به صورت منظم گرداوری کند نمایش می‌دهیم.



شکل 25 - نتایج مدل transformer در حالت mutli step

مدل در پیش‌بینی گام‌های کوتاه‌مدت بسیار موفق عمل کرده و دقت بالایی ارائه داده است. با این حال، با افزایش بازه پیش‌بینی، خطاها به طور قابل توجهی افزایش می‌یابند. این موضوع نشان می‌دهد که مدل نیاز به بهبود در شناسایی وابستگی‌های بلندمدت دارد. به صورت کلی روند کاهش دقت با افزایش زمان پیشبینی قابل مشاهده است که منطقی هم بنظر می‌رسد.

**نتایج مقاله:**

* **t+4**: MAE=58.65، MAPE=0.70%، RMSE=91.80
* **t+8**: MAE=78.50، MAPE=1.12%، RMSE=110.22
* **t+16**: MAE=135.60، MAPE=3.11%، RMSE=166.21

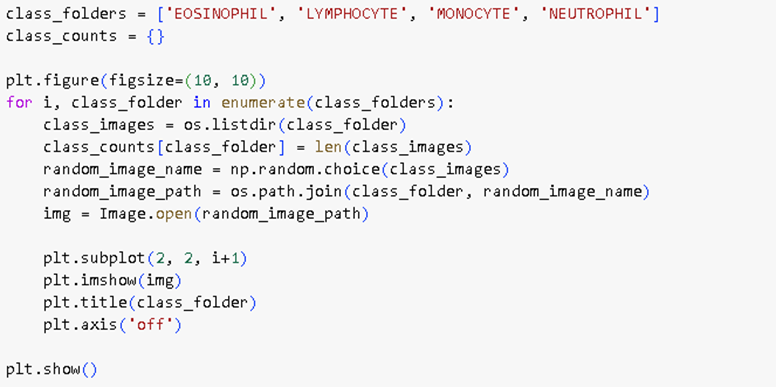
**مقایسه:**

مدل من در کاهش خطاهای MAE و RMSE عملکرد بهتری داشته است. برای مثال، RMSE برای t+4 در مدل من 10.40 و در مقاله 91.80 بوده که بهبود قابل توجهی است. اما MAPE در مقاله برای تمام گام‌ها پایین‌تر گزارش شده است؛ به‌عنوان مثال، MAPE برای t+4 در مدل من 11.48% و در مقاله 0.70% بوده است. این نشان می‌دهد که در مقاله ممکن است استفاده از ورودی‌های متنوع‌تر به کاهش خطای نسبی کمک کرده باشد. البته به احتمال زیاد مقادیر مقاله درصدی گزارش نشده است.

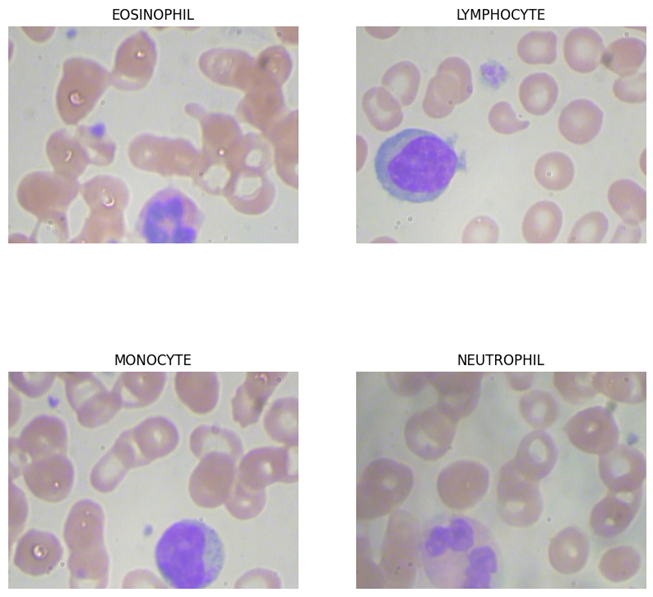
# **پرسش۲. استفاده از ViT برای طبقهبندی تصاویر گلبولهای سفید**

## 1-2. آماده‌سازی داده‌ها

ابتدا از هر کلاس یک تصویر به صورت نمونه نمایش داده­ایم.

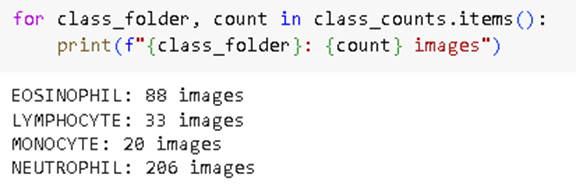


شکل 26- کد نمایش تصادفی یک تصویر از هر کلاس



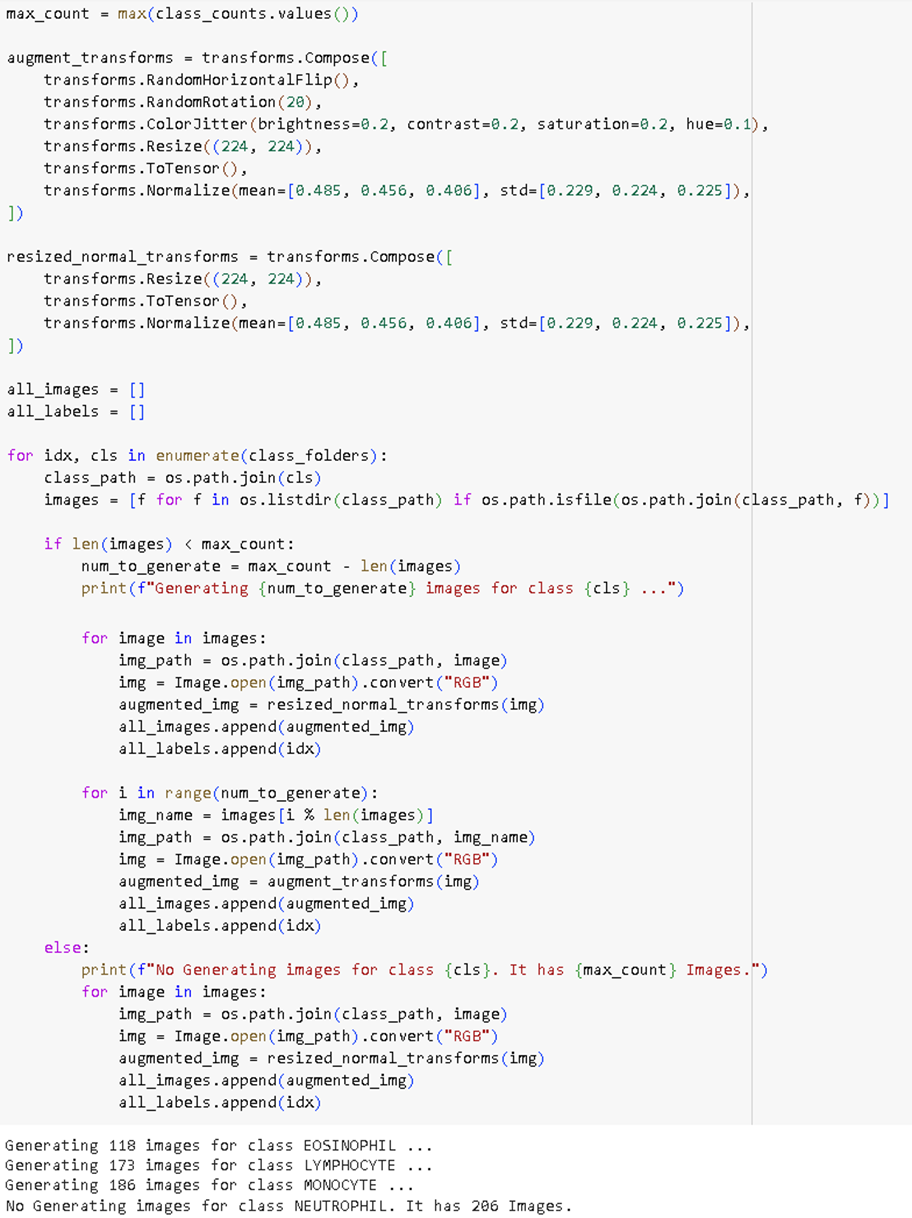
شکل 27- نمایش تصادفی یک تصویر از هر کلاس

سپس از تصاویر هر کلاس را شمرده­ایم که در کلاس EOSINOPHIL تعداد 88، در کلاس LYMPHOCYTE تعداد 33، در کلاس MONOCYTE تعداد 20 و در کلاس NEUTROPHIL تعداد 206 تصویر قرار دارد.



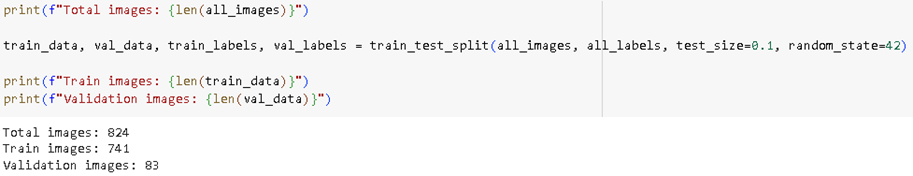
شکل 28- تعداد تصاویر در هر کلاس

وقتی تعداد نمونه‌های موجود در هر کلاس تقریباً برابر باشد، مجموعه داده متوازن است. و وقتی تعداد نمونه‌های یک یا چند کلاس به طور قابل توجه­ی کمتر یا بیشتر از کلاس‌های دیگر باشد، مجموعه داده نامتوازن است. حال با توجه به تعریف و تعداد داده­های هر کلاس به این نتیجه می­رسیم که تعداد داده­های موجود در دیتاست نامتوازن است و باید متوازن شود. تعداد تصاویر هر کلاس را به با تقویت داده به تعداد حداکثر یعنی همان 206 تصویر می­رسانیم.



شکل 29- متوازن کردن کلاس ها با تقویت داده

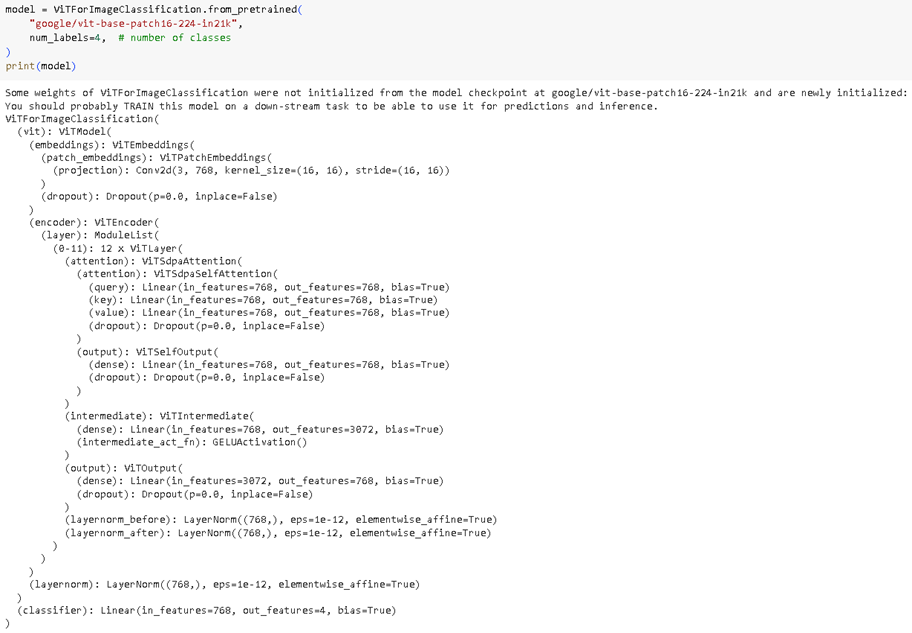
مدل‌های ViT نیاز دارند که تصاویر ورودی به ابعاد خاصی تنظیم شوند، زیرا این مدل‌ها تصاویر را به پچ‌های کوچک تقسیم می‌کنند. ابعاد تصویر معمولاً به صورت H×W مشخص می‌شود، که H و W عرض و ارتفاع تصویر هستند. ابعاد استاندارد 224×224 هستند. برای همین تمام تصاویر را به ابعاد گفته شده تبدیل کرده­ایم.



شکل 30- تقسیم داده به دو قسمت آموزش و ارزیابی

## 2-2. آموزش مدل‌ها

مدل Vision Transformer (ViT) از معماری ترانسفورمر برای تحلیل داده‌های تصویری استفاده می‌کند. برخلاف مدل‌های کانولوشن عصبی (CNN)، این مدل تصاویر را به‌صورت توالی‌هایی از پچ‌ها تقسیم کرده و از مکانیزم توجه (Attention) برای پردازش آن‌ها استفاده می‌کند.ما می‌توانیم از PyTorch و کتابخانه transformers استفاده کنیم تا مدل ViT گوگل را بارگذاری کرده و آن را برای دسته‌بندی 4 کلاس تنظیم کنیم



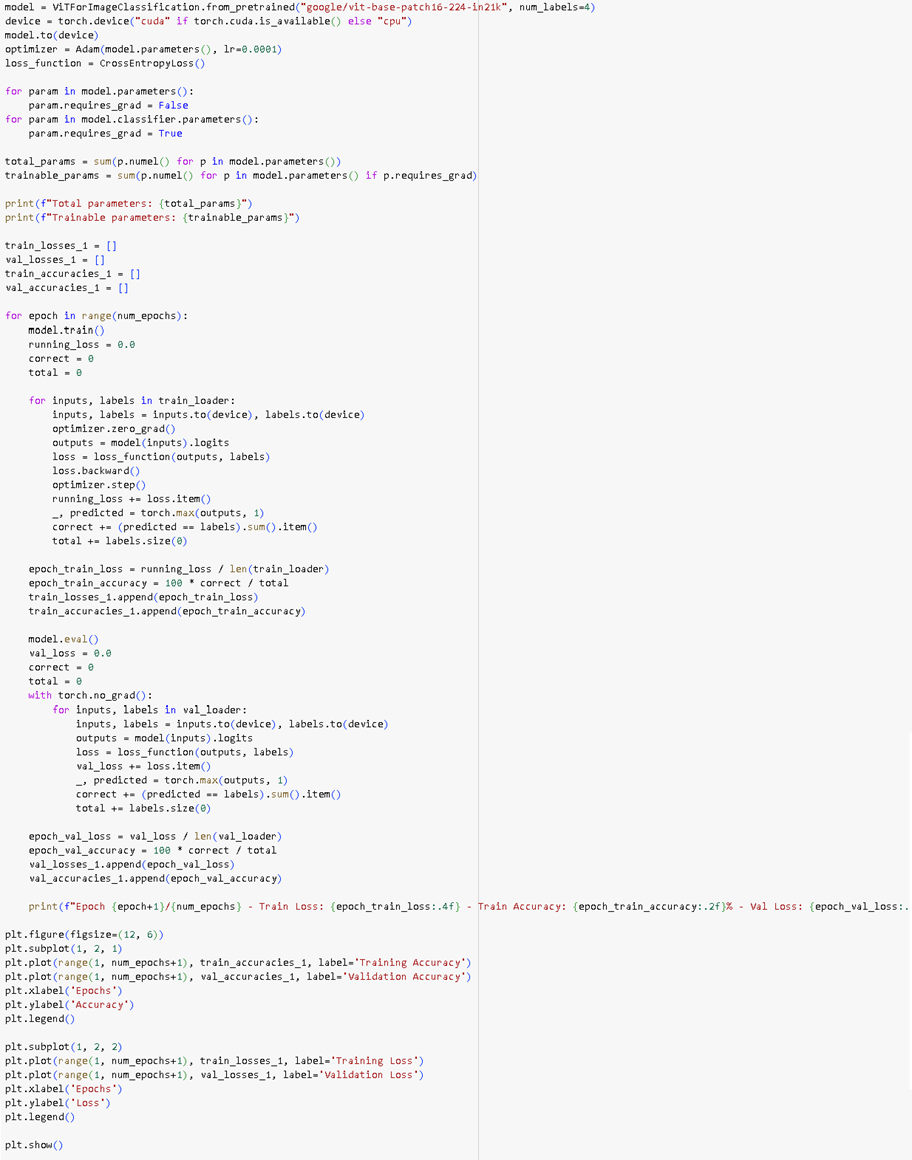
شکل 31- بارگذاری مدل ViT و نمایش معماری آن

خروجی بالا معماری مدل Vision Transformer (ViT) گوگل برای دسته‌بندی تصویر را نشان می‌دهد. این مدل برای 4 کلاس داده تنظیم شده و شامل اجزای زیر است:

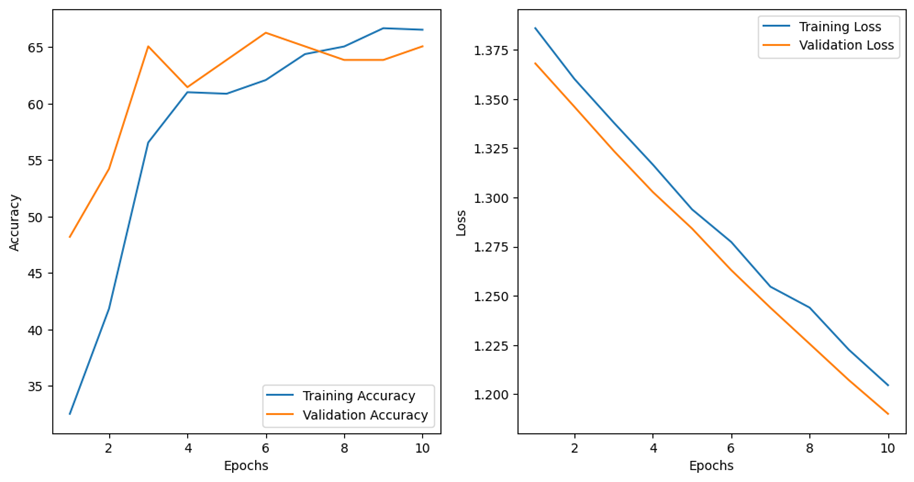
* Embedding: تصاویر ورودی به صورت قطعات یا پچ‌های 16×1616×16 برش داده می‌شوند. هر پچ به کمک یک کانولوشن (Conv2d) به بردارهایی با 768 ویژگی تبدیل می‌شود.
* Encoder: شامل 12 لایه Transformer است که هر کدام دارای موارد زیر هستند:
  + Self-Attention Mechanism: شامل لایه‌های query, key, و value برای محاسبه توجه
  + Intermediate Layer: یک لایه Fully Connected که ورودی را از 768 به 3072 ویژگی گسترش داده و با تابع فعال‌سازی GELU اعمال می‌شود.
  + Output Layer: ورودی 3072 ویژگی را به 768 کاهش می‌دهد.
  + Normalization Layers: هر لایه دارای Normalization قبل و بعد از عملیات پردازشی است.
* LayerNorm: یک لایه نرمال‌سازی نهایی با 768 ویژگی اعمال می‌شود.
* Classifier: یک لایه Fully Connected نهایی که ویژگی‌های خروجی از Encoder را به 4 کلاس دسته‌بندی می‌کند.

این معماری ViT برای یادگیری ویژگی‌های تصویری در حوزه بینایی ماشین بسیار قدرتمند است و با استفاده از مکانیزم توجه (Attention)، روی ویژگی‌های مهم تصویر تمرکز می‌کند.

در حالت اول فقط دسته بند Classifier قابل آموزش است. پس تمام لایه­های مدل را فریز کردیم و فقط Classifier را آزاد گذاشتیم. تعداد کل پارامترها 85801732 و تعداد پارامترهای قابل آموزش 3076 است.

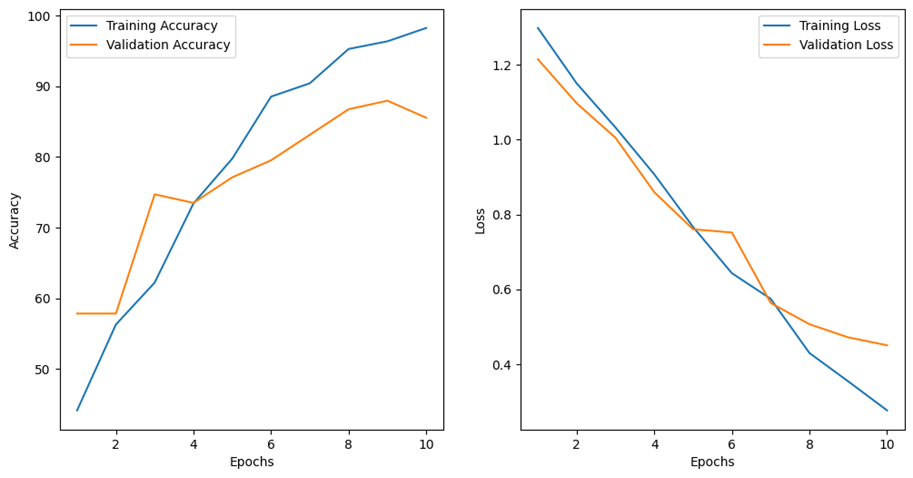


شکل 32- فقط دسته¬بند Classifier قابل آموزش باشد

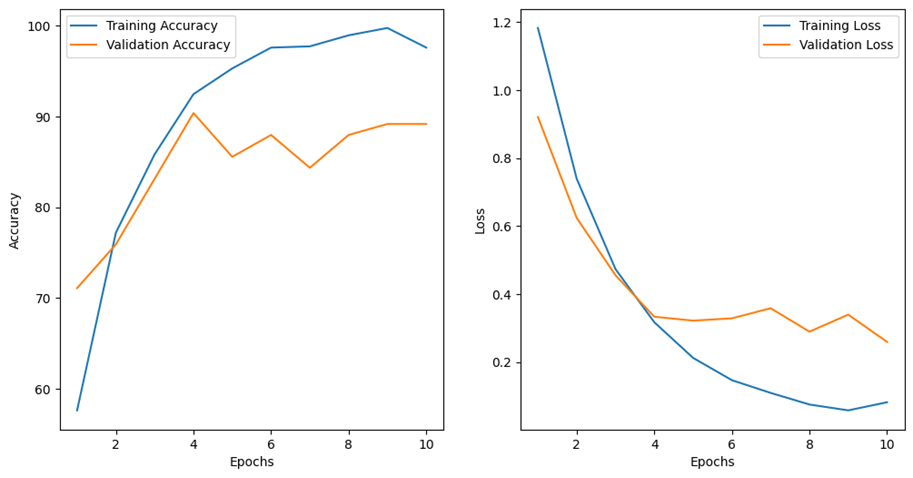


شکل 33- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت یک

در حالت دوم فقط دو لایه اول Encoder قابل آموزش است. لذا همه لایه­ها را ابتدا فریز کردیم و سپس دو لایه اول Encoder و دسته­بند را آزاد گذاشتیم. تعداد کل پارامترها 85801732 و تعداد پارامترهای قابل آموزش 14921476 است.

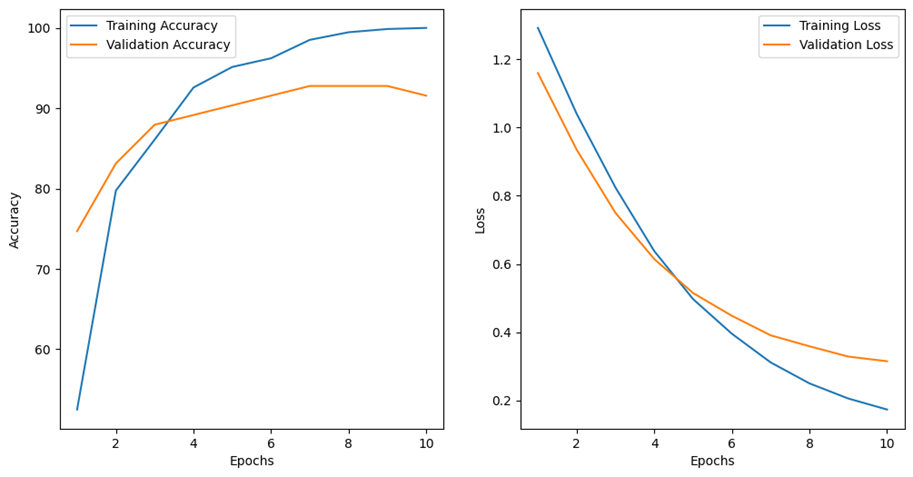


شکل 34- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت دو



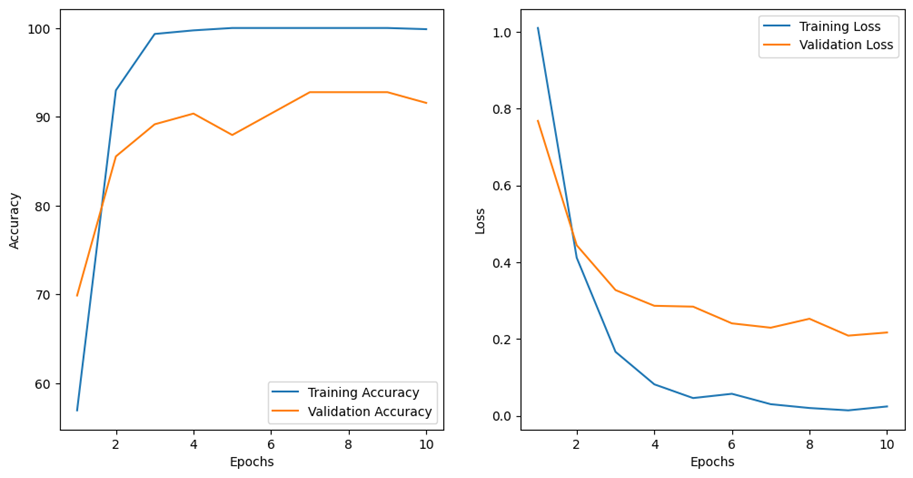
شکل 35- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت سه

در حالت چهارم تمام لایه­ها قابل آموزش است. لذا همه لایه­ها آزاد می­کنیم و مدل را به طور کامل آموزش می­دهیم. تعداد کل پارامترها 85801732 و تعداد پارامترهای قابل آموزش 85801732 است.



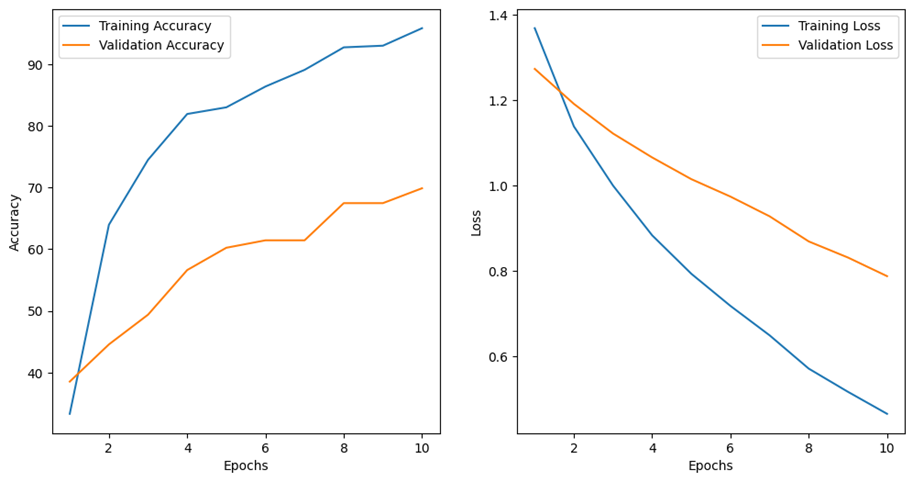
شکل 36- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت چهار

در حالت پنجم یک مدل CNN مثل مدل DenseNet-121 را بارگذاری کردیم و آن را به طور کامل آموزش دادیم. تعداد کل پارامترها 6957956 و تعداد پارامترهای قابل آموزش 6957956 است.



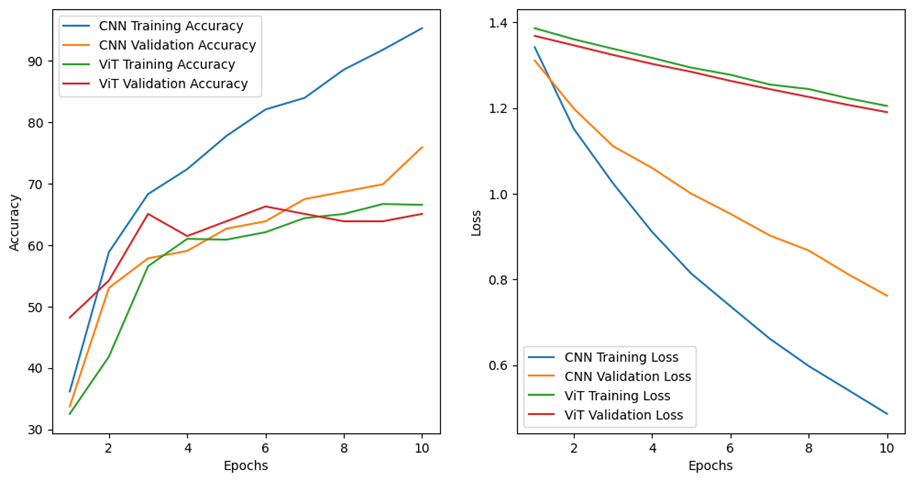
شکل 37- نمودار Accuracy و Loss آموزش و ارزیابی برای حالت پنج

مرحله امتیازی: در حالت ششم یک مدل CNN مثل DenseNet-121 را بارگذاری کردیم و فقط لایه Classifier مدل CNN را آموزش دادیم. تعداد کل پارامترها 6957956 و تعداد پارامترهای قابل آموزش 4100 است.



شکل 38- نمودار Accuracy و Loss برای مدل CCN آموزش فقط در لایه طبقه بند

در آخر نموداری برای مقایسه مدل CNN و ViT که فقط آموزش در لایه Classifiction انجام شده است ترسیم کردیم. با توجه به نمودار بدست آمده واضح است که مدل CNN بهتر آموزش دیده است.



شکل 39- مقایسه مدل CNN و ViT با آموزش در لایه طبقه بند

CNN‌ ها برای استخراج ویژگی‌های محلی (Local Features) طراحی شده‌اند. آنها به خوبی قادر به شناسایی الگوهای محلی در داده‌های تصویری هستند. در داده‌هایی که الگوهای مکانی خاصی دارند (مانند لبه‌ها، خطوط یا الگوهای تکرارشونده)، CNN‌ ها معمولاً عملکرد خوبی نشان می‌دهند. در عوض مدل ViT مبتنی بر مکانیسم توجه (Attention Mechanism) است و تمرکز بیشتری بر ویژگی­های سراسری (Global Features) در تصویر دارد. ViT برای تحلیل تصاویر نیاز به داده‌های بیشتر و پیش‌پردازش مناسب دارد، چرا که ممکن است با داده‌های کوچک و تعداد محدود تصاویر، مدل دچار Overfitting شود و ِا اینکه به خوبی ویژگی­ها را استخراج نکند.

## 3-۲. تحلیل و نتیجه گیری

در این تمرین ما دو مدل CNN و ViT را با حالت­های متفاوت آموزش داده­ایم. با توجه به این که تعداد تصاویر ما در هر کلاس 206 مورد بود عملکرد در مدل شبکه عصب کانولوشنی DenseNet121 بهتر از سایر مدل­ها بود. زیرا آموزش با این مدل در شرایطی که تعداد داده‌ها محدود است، عملکرد بهتری از خود نشان می‌دهد. و در این مدل به خوبی ویژگی‌های محلی را استخراج می­شوند و برای تصاویر با ابعاد کوچک (224\*224) بسیار موثر است.

در مدل ViT به جای استفاده از لایه‌های کانولوشنی، از تکنیک‌های توجه (Attention) برای پردازش تصاویر استفاده می­شود. ViT برای عملکرد مناسب معمولاً به داده‌های آموزشی بیشتری نیاز دارد. ViT به طور کلی برای عملکرد بهینه به حجم داده‌های زیادی نیاز دارد، زیرا مدل نیاز به یادگیری ویژگی‌های پیچیده‌تر و روابط غیرخطی بین بخش‌های مختلف تصویر دارد. در صورتی که تعداد تصاویر محدود است، مدل ViT نتوانسته است به خوبی ویژگی‌های تصویری را استخراج کند، چون به اندازه کافی داده برای آموزش تمام پارامترهایش را نداشته است.