دانشکده مهندسی برق و کامپیوتر

پایان نامه کارشناسی رشته مهندسی کامپیوتر

عنوان پروژه :

تشخیص خودکار علائم راهنمایی و رانندگی با استفاده از یادگیری عمیق

استاد پروژه :

دکتر فاطمه زمانی

ارائه دهنده :

بی تا شاهانی

شهریور ۱۴۰۲

چکیده :

در میان افزایش نگران کننده تصادفات جاده ای و مرگ و میر ناشی از حواس پرتی راننده، خودروسازان به طور فعال برای افزایش ایمنی خودرو تلاش می کنند. این پیگیری منجر به ادغام سیستم‌های پیشرفته کمک راننده (ADAS)با قدرت فناوری‌های هوش مصنوعی (AI) و بینایی کامپیوتری (CV) شده است. یکی از اجزای اصلی این تکامل فناوری، تشخیص علائم ترافیکی (TSR) است، سیستمی که به وسایل نقلیه اجازه می‌دهد تا به طور مستقل طیفی از علائم جاده‌ای، از جمله محدودیت‌های سرعت و علائم توقف را شناسایی و درک کنند. این شاهکار قابل توجه از طریق کاربرد بینایی کامپیوتری و شبکه های عصبی کانولوشن (CNN) در حوزه یادگیری عمیق انجام می شود.

این پیاده سازی یک راه حل کارآمد مبتنی بر CNN برای تشخیص علائم ترافیکی ارائه می دهد که به میزان دقت قابل توجه ۹۷.۸۱ درصد دست می یابد. با استفاده از معماری LeNet، این رویکرد بر محدودیت‌های روش‌های مبتنی بر شکل یا رنگ، که مستعد عوامل محیطی مانند تغییرات نور، مقیاس‌بندی و چرخش هستند، غلبه می‌کند. همچنین معماری شبکه را که با استفاده از کتابخانه‌های Keras و TensorFlow پیاده‌سازی شده است، توضیح می‌دهد و جزئیات فرآیند آموزش دقیق در مجموعه داده‌های معیار تشخیص علائم ترافیک آلمان (GTSRB) را شرح می‌دهد.

نتایج تجربی بر استحکام رویکرد تأکید می‌کند و به دقت قابل توجه ۹۷.۸۱ درصد در تشخیص علائم راهنمایی و رانندگی دست می‌یابد. این کمک به پیشرفت ADAS مبتنی بر هوش مصنوعی، تقویت ایمنی جاده‌ها و پیشرفت چشم‌انداز وسایل نقلیه کاملاً خودمختار کمک می‌کند.

کلمات کلیدی: هوش مصنوعی، شبکه عصبی کانولوشن، یادگیری عمیق، معماریLeNet، تشخیص علائم ترافیکی، Keras، TensorFlow.

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

علائم راهنمایی و رانندگی به سبب آنکه اطلاعات ضروری را برای رانندگان به منظور پیروی از مقررات جاده ای ارائه می دهد، اهمیت زیادی دارد. تشخیص دقیق علائم راهنمایی و رانندگی به جاده‌های ایمن‌تر، کارآمدتر و سازمان‌دهی‌شده‌تر کمک می‌کند و به نفع رانندگان و عابران پیاده است و در عین حال احتمال تصادفات و تراکم ترافیک را کاهش می‌دهد.

شکل ۱-۱ : نمونه ای از علائم راهنمایی و رانندگی

برای شناسایی و طبقه بندی علائم راهنمایی و رانندگی روش های بینایی کامپیوتر قدیمی و نیز روش های مبتنی بر شکل و رنگ نیز استفاده میشد اما این روش ها هم زمانبر بودند و همچنین در مقابل عوامل مختلف مانند تغییر نور، چرخش و مانند ان دچار ضعف بودند.

برای مقابله با این چالش ها، رویکرد های یادگیری ماشین [۱]و یادگیری عمیق[۲] مورد بحث قرار گرفت. به خصوص یادگیری عمیق، به دلیل توانایی ان در طبقه بندی با کیفیت بالا و یادگیری نمایشی از داده های خام مورد توجه قرار گرفت.

یادگیری ماشین زیرشاخه ای از هوش مصنوعی است که به طور کلی به عنوان توانایی ماشین برای تقلید از رفتار هوشمند انسان تعریف می شود. سیستم‌های هوش مصنوعی برای انجام وظایف پیچیده به روشی مشابه نحوه حل مشکلات انسان‌ها استفاده می‌شوند. در برخی موارد، نوشتن برنامه‌ای که ماشین باید آن را دنبال کند زمانبر یا غیرممکن است، مانند آموزش کامپیوتر برای تشخیص تصاویر افراد مختلف. در حالی که انسان ها می توانند این کار را به راحتی انجام دهند ؛ سخت است که به کامپیوتر بگوییم چگونه آن را انجام دهد. یادگیری ماشینی رویکردی را اتخاذ می کند که به رایانه ها اجازه می دهد تا برنامه ریزی خود را از طریق تجربه یاد بگیرند. یادگیری ماشینی همه چیز در مورد آموزش کامپیوترها برای یادگیری از مثال ها و تصمیم گیری بر اساس الگوهایی است که در داده ها کشف می کنند. این مانند هدایت یک کامپیوتر برای بهبود خود با نشان دادن چیزهای مختلف به آن است.
یادگیری ماشینی ارتباط نزدیکی با سایر بخش‌های هوش مصنوعی دارد:

پردازش زبان طبیعی [۳]:

در مورد آموزش ماشین ها برای درک و استفاده از زبان انسان، مانند صحبت کردن و نوشتن است. این به ماشین‌ها کمک می‌کند مانند انسان صحبت کنند و بفهمند ما چه می‌گوییم. درست مانند کاری که الکسا[۴] می کند. الکسا با کمک پردازش زبان طبیعی می تواند حرف ها را بفهمد و مانند یک فرد پاسخ دهد.

شبکه های عصبی [۵]:

شبکه های عصبی مانند مغزهای کامپیوتری هستند که از مغز ما الهام گرفته اند. آنها از بسیاری از قطعات کوچک (گره ها) ساخته شده اند که با هم کار می کنند. گره ها را به عنوان سلول های مغزی در نظر بگیرید که با یکدیگر صحبت می کنند. هر گره یک کار را انجام می دهد، مانند تشخیص یک چشم در یک تصویر. آنها اطلاعات را به یکدیگر منتقل می کنند و تصمیم می گیرند.

---

۱ Machine Learning
۲ Deep Learning
۳ Natural language processing
۴ Alexa
۵ Neural networks

یادگیری عمیق [۶]

یادگیری عمیق مانند داشتن لایه های مغزی زیاد است. هر لایه بخش های مختلف داده را درک می کند. این قدرتمند است اما به نیروی کامپیوتر زیادی نیاز دارد. درست مانند مغز ما برای کارهای مختلف، یادگیری عمیق نیز لایه هایی برای کارهای مختلف دارد، مانند تشخیص چهره در عکس ها. لایه های بیشتر به معنای تصمیم گیری هوشمندانه تر است.[۱]

یادگیری عمیق از مغز انسان الهام می گیرد. در مغز ما سلول های به هم پیوسته ای داریم که برای درک جهان با هم کار می کنند. در یادگیری عمیق، ما چیزی مشابه به نام شبکه های عصبی داریم. این شبکه ها دارای لایه هایی از گره های به هم پیوسته هستند، درست مانند سلول های مغز ما.

شبکه‌های عصبی کانولوشن یک معماری خاص از شبکه‌های عصبی عمیق هستند که در یادگیری عمیق و یادگیری ماشین استفاده می‌شوند. شبکه‌های عصبی کانولوشن [۷]معمولاً برای تصویرسازی بصری استفاده می‌شوند و به رایانه کمک می‌کنند تا تصاویر را شناسایی کرده و از آنها یاد بگیرد. آنها به رایانه بینایی کمک می کنند تا تصویر ورودی را ببینند، آن را طبقه بندی کند، الگوها را ببینند و به طور کلی از آن بیاموزد.[۲]

شبکه‌های عصبی کانولوشن با یادگیری خودکار الگوها و ویژگی‌های سلسله مراتبی از داده‌های پیکسل خام، انقلابی در زمینه بینایی رایانه ایجاد کرده‌اند و نیاز به مهندسی ویژگی‌های دست ساز را کاهش داده‌اند.

هدف این است که با استفاده از یادگیری عمیق و پرداختن به چالش های محیطی، عملکرد سیستم های تشخیص علائم ترافیکی را برای رانندگی ایمن تر بهبود بخشیم. بدین منظور در ادامه ابتدا به پیش پردازش ها می پردازیم سپس اقدام به توضیح معماری شبکه عصبی کانولوشن میکنیم سپس به پباده سازی مدل خواهیم پرداخت. ما این مدل را به زبان پایتون پیاده سازی خواهیم کرد.

---

[۶] **Deep learning**

[۷] CNN

فصل ۲

پیش پردازش

پیش پردازش تصاویر گامی مهم در شبکه های عصبی کانولوشن (CNN) است زیرا زمینه را برای یادگیری موثر و پیش بینی های دقیق فراهم می کند. به عبارت دیگر پیش پردازش زمینه را برای شبکه های عصبی کانولوشن فراهم می کند تا به طور موثر یاد بگیرد، پیش بینی های دقیق انجام دهد و چالش های دنیای واقعی را با اطمینان مدیریت کند.

در مرحله ی پیش پردازش ما چهار گام را باید انجام دهیم :

در گام نخست همانطور که پیش تر گفته شد تصاویر ما به حالت رنگی (RGB) می باشد و باید ان ها را به سطح خاکستری (Gray Scale) تبدیل نماییم،

گام بعد اعمال یکسان سازی هیستوگرام (Histogram Equalization) به تصاویر می باشد،

در گام سوم باید تصاویر را نرمال سازی کنیم،

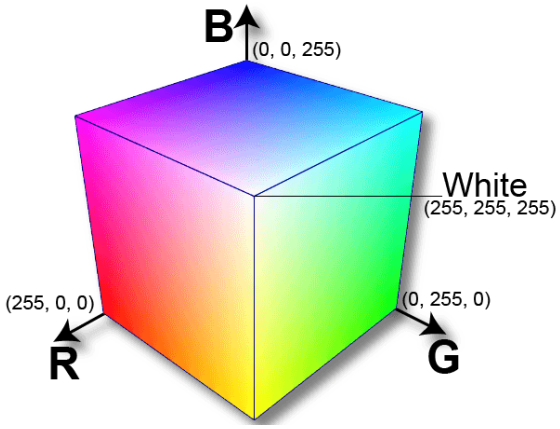
و نهایتا در گام اخر افزایش داده را انجام می دهیم.

در ادامه به هر یک از این گام ها می پردازیم.

۱-۲ تبدیل تصویر رنگی به خاکستری

RGB شناخته شده ترین مدل رنگ است که مخفف سه رنگ قرمز (RED)، سبز (GREEN) و آبی (BLUE) می باشد. قرمز و سبز و آبی را می توان در نسبت های مختلف ترکیب کرد تا هر رنگی در طیف رنگی بدست اید.

یک رنگ با استفاده از سه مقدار صحیح از صفر تا ۲۵۵ برای سه رنگ قرمز و سبز و ابی تعریف می شود که صفر به معنای تیره و ۲۵۵ به معنای روشن می باشد. این مدل تقریبا در تمامی صفحات دیجیتال در سراسر جهان استفاده می شود. دستگاه هایی مانند صفحه نمایش و دوربین ها از RGB برای نمایش و گرفتن رنگ ها استفاده می کنند. پیکسل ها این سه رنگ را ساطع می کنند یا می گیرند تا تمام رنگ هایی را که می بینید ایجاد کنند.



شکل ۱-۲ : مدل رنگی RGB

مقیاس خاکستری (Grayscale) ساده ترین مدل است زیرا رنگ ها را تنها با استفاده از یک جزء بیان می کند و مقدار روشنایی با استفاده از مقداری از صفر(سیاه) تا ۲۵۵ (سفید) توصیف می شود.

تصاویر در خاکستری یک نمایش ثابت در شرایط مختلف نور و تنوع رنگ ارائه می دهد که این می تواند قابلیت تعمیم مدل را بهبود بخشد و آن را نسبت به تغییرات رنگ کمتر حساس کند. همچنین استفاده از از این مقیاس به فضای کمتری نیاز دارد و سریعتر است؛ به خصوص زمانی که با محاسبات پیچیده سر و کار داریم زیرا آن ها سه برابر داده کمتر از تصاویر رنگی دارند.[۳]

بنابراین به طور کلی، تبدیل از RGB به خاکستری در مرحله پیش پردازش مدل CNN می‌تواند با تمرکز بر ویژگی‌های ساختاری مهم تصاویر و در عین حال کاهش پیچیدگی محاسباتی و استفاده از حافظه، منجر به آموزش ساده‌تر، سریع‌تر و بالقوه موثر تر شود.

تبدیل تصویر رنگی یا همان RGB به تصویر سطح خاکستری یا همان Grayscale با ترکیب وزنی زیر از سه رنگ اصلی قرمز و سبز و ابی تشکیل می شود:

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

شکل ۲-۲ : ترکیب وزنی تبدیل تصویر رنگی به سطح خاکستری

برای این تبدیل رنگی در پایتون در ابتدا کتابخانه ی زیر را می خوانیم:

```
import cv۲
```

سپس با استفاده از تابع زیر:

```
Cv۲.cvtColor("your image", cv۲.COLOR_BGR۲GRAY)
```

این تبدیل انجام می شود.

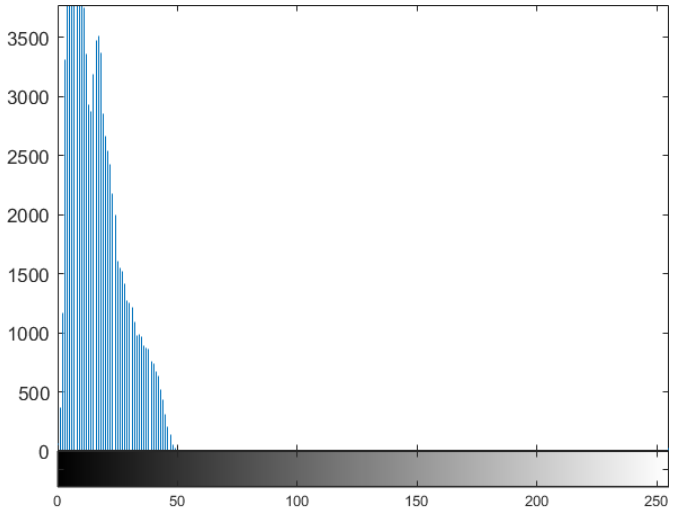
۲-۲ یکسان سازی هیستوگرام

هیستوگرام راهی برای بهبود کنتراست و تعادل تصویر است. این مانند تنظیم سطوح روشنایی برای برجسته کردن جزئیات است. این یک راه ساده برای افزایش کنتراست و بهبود کیفیت بصری تصاویر است.
تصویر ارایه ای از پیکسل ها است و همانطور که پیش تر گفته شد پیکسل ها می توانند مقادیری از صفر تا ۲۵۵ را شامل شوند و هیستوگرام تصویر توزیع مقادیر پیکسل های تصویر می باشد.

به عبارت دیگر اگر تصویر زیر را در نظر گرفت:

شکل ۳-۲ : تصویر قبل اعمال یکسان سازی هیستوگرام

که نمودار هیستوگرام ان به شکل زیر می باشد:



شکل ۴-۲ : نمودار هیستوگرام تصویر

در نمودار فوق محور افقی نشان دهنده ی شدت پیکسل ها می باشد و محور عمودی بیانگر فراوانی ان ها است. با دقت به نمودار حاصل و شکل اولیه مشخص است که فراوانی پیکسل های تاریک تصویر زیاد است و همین امر سبب شده است جزئیات روشن تصویر به خوبی دیده نشود و تصویر تاریک دیده شود.

همانطور که از نمودار هیستوگرام شکل مشاهده شد، بیشتر پیکسل ها دارای شدت بین ۰ تا ۵۰ هستند، درواقع یعنی قسمت کوچکی از تصویر که چشم انسان می تواند ان را سفید ببیند هم تقریبا سیاه دیده می شود.

یا می توان گفت قسمت های سیاه تصویر، سیاه است و قسمت های سفید تصویر، کمتر سیاه است.

کاری که یکسان سازی هیستوگرام در این تصویر انجام می دهد ان است که شدت قسمت های سفید تصویر را افزایش داده، بدون انکه روی قسمت های سیاه تصویر تاثیری بگذارد.[٤]

بدین صورت بعد از اعمال یکسان سازی هیستوگرام تصویر بالا به شکل زیر دیده می شود:

شکل ۵-۲ : تصویر بعد از اعمال یکسان سازی هیستوگرام

در اصل، یکسان سازی هیستوگرام توزیع مقادیر شدت را تنظیم می کند تا تصویر از نظر بصری جذاب تر و آموزنده تر شود. این تکنیک به طور گسترده در کاربردهای مختلف مانند تصویربرداری پزشکی، تصاویر ماهواره‌ای و بینایی رایانه‌ای برای بهبود کیفیت تصویر و بهبود ویژگی‌هایی استفاده می‌شود که در غیر این صورت تشخیص آنها دشوار است.

برای عملیات یکسان سازی هیستوگرام در پایتون بعد از خواندن کتابخانه های مرد نیاز:

```
import numpy as np
from skimage import io, exposure
```

با استفاده از تابع زیر این عملیات یکسان سازی هیستوگرام را برای تصاویر خود انجام می دهیم:

```
histogramImage = exposure.equalize_hist("your image")
```

۲-۳ نرمالسازی تصاویر

همانطور که پیش تر گفته شد محدوده ی مقادیر پیکسل های تصویر بین ۰ تا ۲۵۵ می باشد .

وقتی از این تصاویر در یک شبکه ی عصبی عمیق استفاده می کنیم، محاسبات می توانند به دلیل این مقادیر پیکسل بزرگ بسیار پیچیده و کند شوند.

بنابراین برای ساده تر و سریع تر کردن کار ها در شبکه های عصبی، می توانیم مقادیر پیکسل را نرمال کنیم.

به این معنا که محدوده ی ۰ تا ۲۵۵ را به یک محدوده ی جدید صفر تا یک تبدیل می کنیم و این کار با تقسیم تمام مقادیر پیکسل بر ۲۵۵ صورت می گیرد.[۵]

بدون نرمال سازی، مقادیر بسیار زیاد یا پایین پیکسل ممکن است بر فرآیند یادگیری غالب شود. عادی سازی از تحت الشعاع قرار دادن این موارد شدید دیگر ویژگی ها جلوگیری می کند. عادی سازی تصاویر در پیش پردازش مانند اطمینان از متعادل و متناسب بودن تمام مواد تشکیل دهنده یک دستور است. با مقیاس بندی مقادیر پیکسل به یک محدوده ثابت، به CNN کمک می کند تا به طور موثر یاد بگیرد، سریع تر همگرا شود و تغییرات در داده ها را بدون غرق شدن مدیریت کند.
بدین ترتیب برای نرمال سازی تصویر در پایتون به صورت زیر عمل می کنیم:

```
normalize image  = “your image” / ۲۵۵
```

نرمالسازی مقادیر را به یک محدوده ثابت کاهش می دهد، کارایی یادگیری را افزایش می دهد و داده ها را برای الگوریتم های یادگیری عمیق قابل مدیریت تر می کند.

شکل ۶-۲ : نمونه ای از تصویر قبل و بعد از نرمال سازی

۲-۴ افزایش داده

افزایش داده های تصویر تکنیکی است که تصاویر جدیدی از تصاویر موجود ایجاد می کند. برای انجام این کار، تغییرات کوچکی مانند تنظیم روشنایی تصویر، چرخاندن تصویر[۸]، یا جابجایی[۹] سوژه در تصویر به صورت افقی یا عمودی در آنها ایجاد می شود.
تکنیک های افزایش تصویر این امکان را می دهد که به طور مصنوعی اندازه مجموعه آموزشی خود را افزایش داده، در نتیجه داده های بسیار بیشتری را برای مدل خود برای آموزش فراهم کنیم. که این امکان را می دهد که دقت مدل خود را با افزایش توانایی مدل خود در تشخیص انواع جدید داده های آموزشی خود بهبود بخشیم[٦].
تقویت داده ها در یک مدل شبکه عصبی کانولوشن (CNN) مانند افزودن تغییراتی به تصاویر آموزشی است تا مدل را قوی تر و در مدیریت موقعیت های مختلف بهتر کند. مثل این است که به مدل، نسخه های کمی متفاوت از یک تصویر را نشان دهید تا به یادگیری بهتر آن کمک کنید.

از انجایی که در خیابان ها، علائم راهنمایی و رانندگی ممکن است چرخانده شوند یا شیب داشته باشند و

یا مواردی مانند ان، برای بهبود بیشتر عملکرد مدل، از تکنیک افزایش داده استفاده می شود.

به طور مثال یک علامت توقف را تصور کنید که کمی کج شده است یا در شرایط نوری مختلف ثبت شده است. افزایش داده ها به مدل ارائه شده این امکان را می دهد تا این علائم را به درستی تشخیص داده و تفسیر نماید.

این افزایش داده شامل ایجاد نسخه های جدید از تصاویر در مجموعه داده اموزشی با ایجاد تغییراتی مانند جا به جایی (shift)، بزرگنمایی (zoom) و چرخش (rotation) در تصاویر می باشد.

افزایش داده ها تکنیکی است در یادگیری عمیق برای افزایش مصنوعی تنوع مجموعه داده اموزشی استفاده

می شود.

[۸] rotating
[۹] shifting

برای اعمال افزایش داده در داده های علائم راهنمایی و رانندگی و در زبان پایتون به شکل زیر عمل می کنیم:

ابتدا کتابخانه ی مورد نیاز زیر را می خوانیم :

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

در گام بعد داده های ورودی را به شکل زیر تغییر شکل دادیم:

```
#Reshape the input data for grayscale image
images = images.reshape(-1, 32 , 32 , 1)
```

شکل ۷-۲ : تغییر شکل داده های ورودی برای افزایش داده

CNN ها انتظار دارند که تصاویر شکل خاصی داشته باشند که بتوانند به طور موثر پردازش کنند. این شکل شامل ارتفاع، عرض، کانال ها و گاهی اوقات اندازه دسته است.

از تغییر شکل برای برای تنظیم ساختار داده ها، برای مطابقت با آنچه CNN ها انتظار دارند استفاده می شود.

در این مرحله تصاویر به گونه ای تغییر شکل داده ایم که چهار بعد داشته باشند : بعد اول نشان دهنده ی تعداد تصویر، بعد دوم و سوم نشان دهنده ی طول و عرض تصویر و بعد سوم نشان دهنده ی تعداد کانال ها (که ۱ کانال بیانگر همان سطح خاکستری می باشد) است.

استفاده از ۱- این امکان را می دهد تا ارایه را بدون تعیین صریح تعداد تصاویر تغییر شکل دهیم درواقع «۱-» به عنوان یک مکان نگه‌دار هوشمند عمل می‌کند و این امکان را می‌دهد که بدون نگرانی در مورد محاسبات ابعاد دقیق، آرایه‌ها را تغییر شکل دهیم و کد را سازگارتر و مدیریت آن آسان‌تر می‌کند.

زیبایی استفاده از «۱-» این است که اطمینان حاصل می کند که تمام عناصر موجود در آرایه به درستی استفاده و تغییر شکل داده شده اند.

در گام بعد ImageDataGenerator را با پارامتر ها به شکل زیر تعریف می کنیم:

```
# Data augmentation using ImageDataGenerator
data_augmentor = ImageDataGenerator(
    rotation_range=10, # Randomly rotate images by 10 degrees
    width_shift_range=0.1, # Randomly shift images horizontally by 10% of the width
    height_shift_range=0.1, # Randomly shift images vertically by 10% of the height
    zoom_range=0.2 # Randomly zoom into images by 20%
)
```

شکل ۸-۲ : پیکربندی با پارامتر ها برای افزایش داده ها

معین کردیم که تصاویر ۱۰ درجه تصادفی به هر جهت بچرخد و ده درصد به صورت افقی و عمودی جا به جا کند و تا بیست دصد روی تصاویر زوم کند.

در گام بعد با استفاده از تابع train_test_split() که از کتابخانه ی scikit_learn می باشد، داده های اموزشی و ازمایشی را از هم جدا می کنیم به طوریکه ۲۰ درصد از داده ها را به عنوان ازمایش و ۸۰ درصد از داده ها را به عنوان اموزش در نظر میگیریم. این کار کمک می کند تا حدس بزنیم که مدل ما چقدر به داده های جدید و دیده نشده تعمیم می یابد. به عبارت دیگر در مدل های یادگیری مهم است که عملکرد مدل را بر روی داده هایی که در طول آموزش ندیده اند، ارزیابی کرد.

```
# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)
```

شکل ۹-۲ : تقسیم داده ها به مجموعه ی اموزشی و ازمایشی

در آن x_train و y_train به ترتیب مجموعه ی تصاویر اموزشی و ازمایشی می باشند و x_test و y_test به ترتیب مجموعه ی برچسب های اموزشی و ازمایشی می باشند و "Random_state=٤٢" مانند یک عدد خاص است که کد را در اجراهای مختلف ثابت نگه می‌دارد.

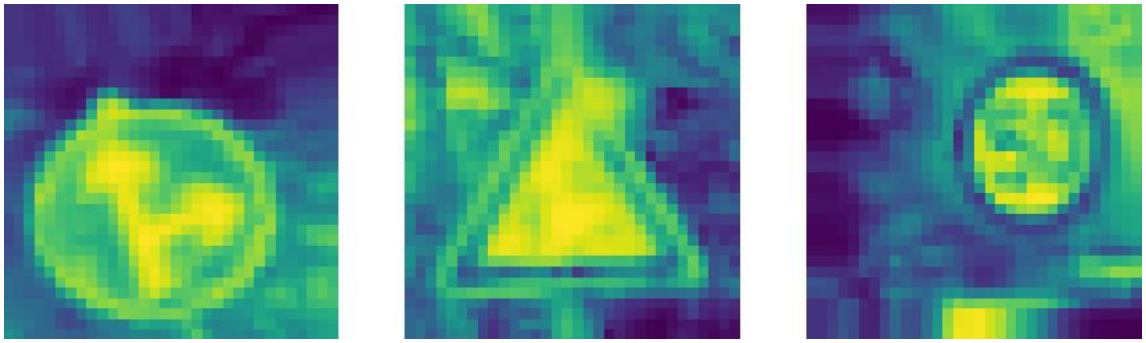
سپس در گام اخر تنظیمات ارائه شده در بالا را برای افزایش داده ها به مجموعه ی تصاویر اموزشی خود به شکل زیر اعمال کردیم:

```
# Fit the ImageDataGenerator on training data
data_augmentor.fit(X_train)
```

شکل ۱۰-۲ : اعمال افزایش داده بر اساس پارامتر های داده شده

باتوجه به تنظیمات مشخص شده در مرحله ی قبل تبدیل های تصادفی به تصاویر اموزشی اعمال می شود و داده های جدید تولید می شود.

در زیر برخی از این تصاویر بعد از اعمال افزایش داده دیده می شود:



شکل ۱۱-۲ : نمونه ای از تصاویر بعد از افزایش داده

فصل ۳

معماری شبکه عصبی کانولوشن

با پیشرفت های اخیر شبکه های عصبی کانولوشن در حوزه ی بینایی کامپیوتر، مدل های معروفی از این شبکه ها به وجود امدند. در اصل، هر معماری به چالش خاصی می پردازد یا ایده جدیدی را برای افزایش قابلیت های CNN ها بررسی می کند. با گذشت زمان، این تنوع معماری منجر به درک بهتر اصول طراحی شبکه های عصبی شده و راه را برای پیشرفت در زمینه بینایی کامپیوتر هموار کرده است. چند نمونه از انواع معماری CNN در جدول ۱-۳ در صفحه ۲۴ آورده شده است.

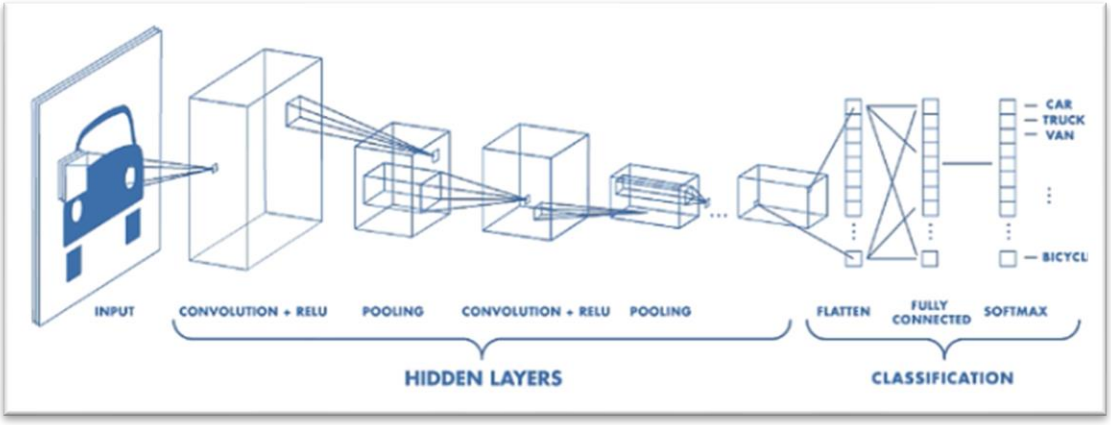
LeNet که به نام سازنده آن Yann LeCun نامگذاری شده است، یک شبکه عصبی کانولوشن است که از چندین لایه تشکیل شده است که برای پردازش تصاویر به صورت سلسله مراتبی طراحی شده است و ویژگی های پیچیده تر را در هر لایه ثبت می کند. این یک پیشرفت قابل توجه در زمینه بینایی کامپیوتر بود و پایه و اساس رویکردهای یادگیری عمیق مدرن در تشخیص تصویر را ایجاد کرد.
مدلی که ما استفاده کردیم نیز بر اساس معماری LeNet است.

معماری شبکه ی عصبی کانولوشن (CNN[۱۰])، که برای تشخیص تصویر از ان استفاده می کنیم، شامل سه بخش اصلی می باشد:

- لایهٔ کانولوشن [۱۱]برای استخراج ویژگی تصویر
- لایهٔ ادغام [۱۲]، برای کاهش ابعاد تصویر با حفظ ویژگی های مهم
- لایهٔ تمام متصل [۱۳](یا dense) برای پیش بینی یا طبقه بندی تصویر ورودی

[۱۰] Convolutional Neural Network
[۱۱] convolutional layer
[۱۲] pooling layer
[۱۳] fully connected layer

شبکه های عصبی کانولوشن به دلیل نتایج عالی و یادگیری خودکار ویژگی ها برای تشخیص تصویر ترجیح داده می شوند.



شکل ۱-۳ : مثال از معماری یک CNN

جدول ۱-۳ : معماری های مختلف CNN

معماری	سال	سازنده	دستاورد مهم
LeNet	۱۹۹۸	Yann LeCun	CNN های پیشگام برای تشخیص دست خط.
AlexNet	۲۰۱۲	Alex Krizhevsky	برنده چالش تشخیص تصویری در مقیاس بزرگ ImageNet (ILSVRC)
GoogleNet	۲۰۱٤	Google AI	ماژول های اولیه را برای بهبود کارایی و دقت معرفی کرد.
VGGNet	۲۰۱٤	Visual Geometry Group	تأثیر عمق شبکه بر عملکرد را بررسی کرد.
ResNet	۲۰۱٥	Microsoft Research	اتصالات باقیمانده را معرفی کرد که امکان آموزش شبکه های بسیار عمیق را فراهم می کند.

۳-۱ لایهٔ کانولوشن

لایهٔ کانولوشن بخش مهمی از شبکه عصبی کانولوشن است و بار محاسباطی شبکه را حمل می کند.

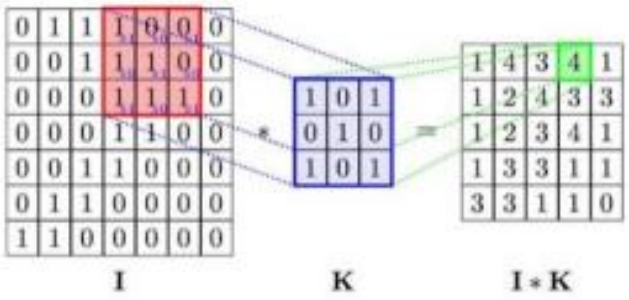
تصویر ورودی را به صورت شبکه ای از پیکسل ها در نظر میگیریم که در این صورت لایه ی کانولوشن از یک شبکه ی کوچکتر از اعداد به نام هسته[۱۴] یا فیلتر استفاده می کند که مانند یک پنجره ی کوچک است که روی تصویر می لغزد.

برای هر موقعیت هسته روی تصویر، لایه کانولوشن اعداد موجود در هسته را با مقادیر پیکسل مربوطه از تصویر ضرب می کند و نتایج را جمع می کند. نتیجه یک عدد واحد است که نشان می دهد که هسته چقدر با یک الگوی خاص در آن قسمت از تصویر مطابقت دارد.

سپس هسته به موقعیت بعدی می رود و فرآیند را تکرار می کند و یک شبکه جدید از اعداد به نام نقشه فعال-سازی ایجاد می کند که نشان می دهد که هسته چقدر با الگوهای مختلف در سراسر تصویر مطابقت دارد.

اندازه حرکت کرنل هنگام لغزش روی تصویر، stride نامیده می شود و به اضافه کردن پیکسل ها به اطراف مرز یک تصویر یا نقشه فعال سازی[۱۵]، قبل از اعمال عملیات کانولوشن padding گفته می شود.

با استفاده از چندین کرنل در لایه کانولوشن، شبکه می تواند یاد بگیرد که ویژگی ها و الگوهای مختلف را در تصویر ورودی تشخیص دهد.[۷]



شکل ۳-۲ : شرح عملیات کانولوشن

[۱۴] kernel
[۱۵] Activation map

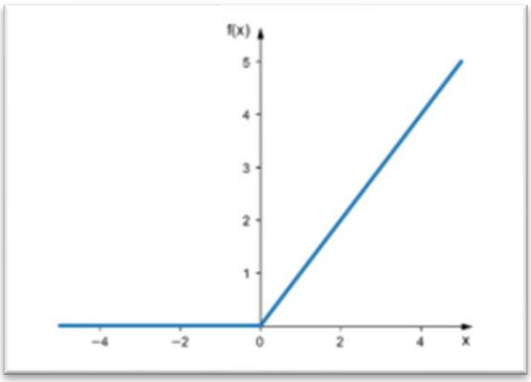
در یک شبکه ی عصبی کانولوشنال، توابع غیرخطی معرفی می شوند تا شبکه را قادر به درک الگوهای پیچیده در تصویر کند.

در اصل، توابع غیرخطی مانند ابزارهای هنری هستند که به CNN ها اجازه می دهند جوهر واقعی تصاویر را ثبت کنند. آنها به CNN ها توانایی درک و پردازش ویژگی ها، اشکال و بافت هایی را می دهند که دنیای داده های بصری را می سازند. با پذیرش این توابع منحنی و منعطف، CNN ها می توانند به طرز ماهرانه ای معمای تصاویر را رمزگشایی کنند و آنها را به گونه ای زنده کنند که روابط ساده و خطی نمی توانند.

در معماری CNN توابع غیر خطی[۱۶] معمولا پس از هر لایه ی کانولوشن معرفی می شوند. بنابراین پس از پردازش لایه ی کانولوشن تصویر و استخراج ویژگی های مربوطه، تابع غیر خطی وارد عمل می شود.

توابع فعالسازی غیر خطی در شبکه های عصبی انواع مختلفی دارد که توابع غیر خطی استفاده شده در مدل ما، ReLU و Softmax می باشد.

تابع فعالسازی غیر خطی ReLU، یکی از رایج ترین توابع فعالسازی شبکه های عصبی کانولوشن محسوب می شود. در این تابع چنانچه ورودی تابع بزرگتر از صفر باشد، خروجی برابر با ورودی تابع و اگر ورودی تابع مقداری کمتر از صفر باشد، خروجی تابع برابر صفر خواهد شد.



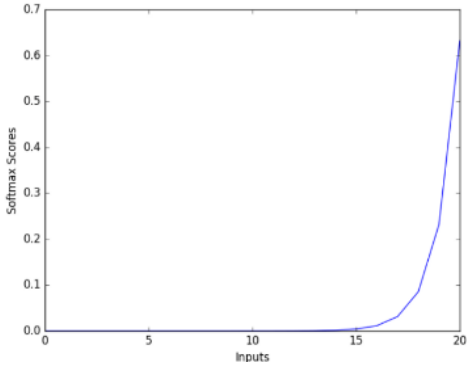
شکل ۳-۳ : نمودار تابع ReLU

[۱۶] Non-Linear Activation Functions

تابع فعالسازی Softmax نوعی از تابع سیگموئید[۱۷] به حساب می اید. به عبارتی ترکیبی از چند تابع سیگموئید است که احتمالات نسبی را محاسبه می کند.درواقع احتمالات چندین کلاس را مشخص می کند.

معمولا در دسته بندی های چندکلاسه از از این تابع فعالسازی در لایه ی اخر استفاده می شود.

این تابع علاوه بر نگاشت خروجی بین صفر و یک، هر خروجی را به نحوی نگاشت می کند که مجموع ان برابر با یک باشد و خروجی این تابع یک توزیع احتمالی است.[۸]

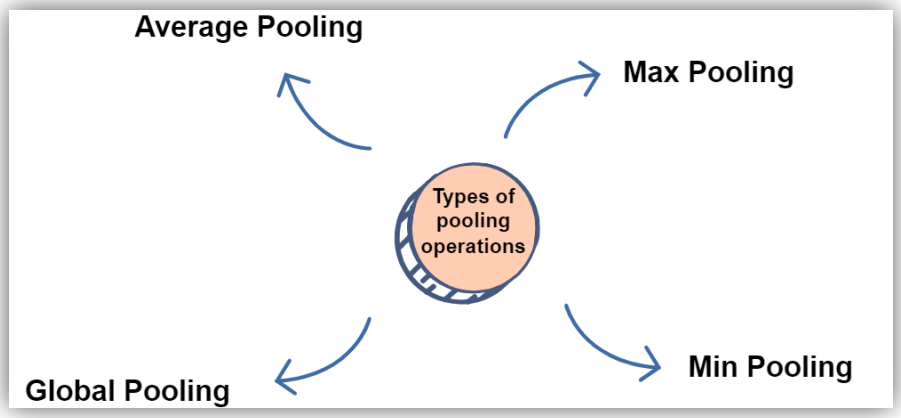


شکل ۴-۳ : نمودار تابع Softmax

[۱۷] تابع فعال سازی سیگموئید یک تابع غیر خطی ساده و پرکاربرد است که مقادیر ورودی را به احتمالات بین ۰ و ۱ ترسیم می کند. این تابع برای کارهای طبقه بندی باینری و افزودن غیرخطی به شبکه های عصبی مفید است، اما اشباع آن در مقادیر شدید می تواند منجر به چالش های آموزشی در شبکه های عمیق شود.

۳-۲ لایهٔ ادغام

لایه های ادغام یکی از بلوک های سازنده شبکه های عصبی کانولوشن هستند. در جایی که لایه‌های کانولوشن، ویژگی‌ها را از تصاویر استخراج می‌کنند، لایه‌های Pooling ویژگی‌های آموخته‌شده توسط CNN را یکپارچه می‌کنند. هدف آن کوچک کردن تدریجی بعد فضایی نمایش برای به حداقل رساندن تعداد پارامترها و محاسبات در شبکه است. مزیت استفاده از pooling آن است که اندازه را کاهش می دهد و نمایش های کوچکتر به حافظه و محاسبات کمتری نیاز دارد و شبکه را سریع تر و کارامدتر می کند و نیز به شبکه کمک می کند تا با دور انداختن اطلاعات کمتر مرتبط، بر ویژگی های مهم تر تمرکز کند.



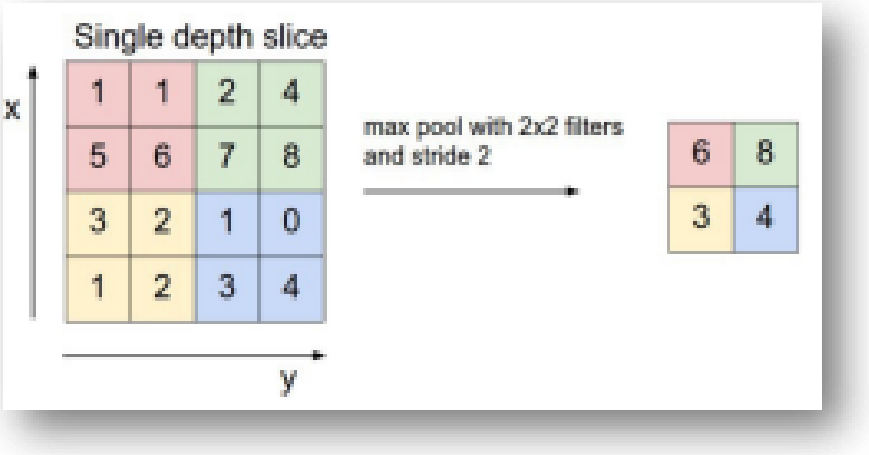
شکل ۵-۳ : انواع عملیات ادغام

دو نوع متداول لایه های ادغام وجود دارد: Max Pooling و Average Pooling.

Max Pooling : در این نوع ادغام، خلاصه ویژگی های یک منطقه با حداکثر مقدار در آن ناحیه نشان داده می شود. بیشتر زمانی استفاده می‌شود که تصویر پس‌زمینه تیره داشته باشد، زیرا حداکثر ادغام پیکسل‌های روشن‌تری را انتخاب می‌کند و تصویر برگشتی واضح تر از تصویر اصلی می باشد.

Average Pooling : این لایه ترکیبی با بدست آوردن میانگین کار می کند. میانگین ادغام مقادیر متوسط ویژگی های نقشه ویژگی را حفظ می کند. این تصویر را صاف می کند در حالی که ماهیت ویژگی را در یک تصویر حفظ می کند. این به ما نسخه‌ای نرم‌تر و آرام‌تر از تصویر می‌دهد که در آن هر قسمت میانگینی از همسایگان خود را نشان می‌دهد.

به طور کلی لایه ی pooling از اجزای حیاتی شبکه های کانولوشنی می باشد که به کاهش محاسبات، حفظ ویژگی های مهم و بهبود عملکرد شبکه در تشخیص تصویر کمک می کند.[۹]



شکل ۶-۳ : نمونه عملیات max pooling

۳-۳ لایهٔ تمام متصل

معمولا آخرین لایه های یک شبکه عصبی کانولوشن برای طبقه بندی را لایه های **تمام متصل**[18] تشکیل می‌دهند.

یکی از کاربردهای اصلی لایهٔ تمام متصل در شبکه کانولوشن، استفاده به عنوان طبقه بند[۱۹] است. یعنی مجموعه ویژگی‌های استخراج شده با استفاده از لایه های کانولوشنی درنهایت تبدیل به یک بردار می‌شوند. درنهایت این بردار ویژگی به یک طبقه بند تمام متصل داده می‌شود تا کلاس درست را شناسایی کند[۱۰].

لایهٔ ورودی تمام متصل[۲۰]: خروجی لایه های قبلی را می گیرد، آنها را "مسطح" می کند و آنها را به یک بردار تبدیل می کند که می تواند ورودی برای مرحله بعدی باشد.

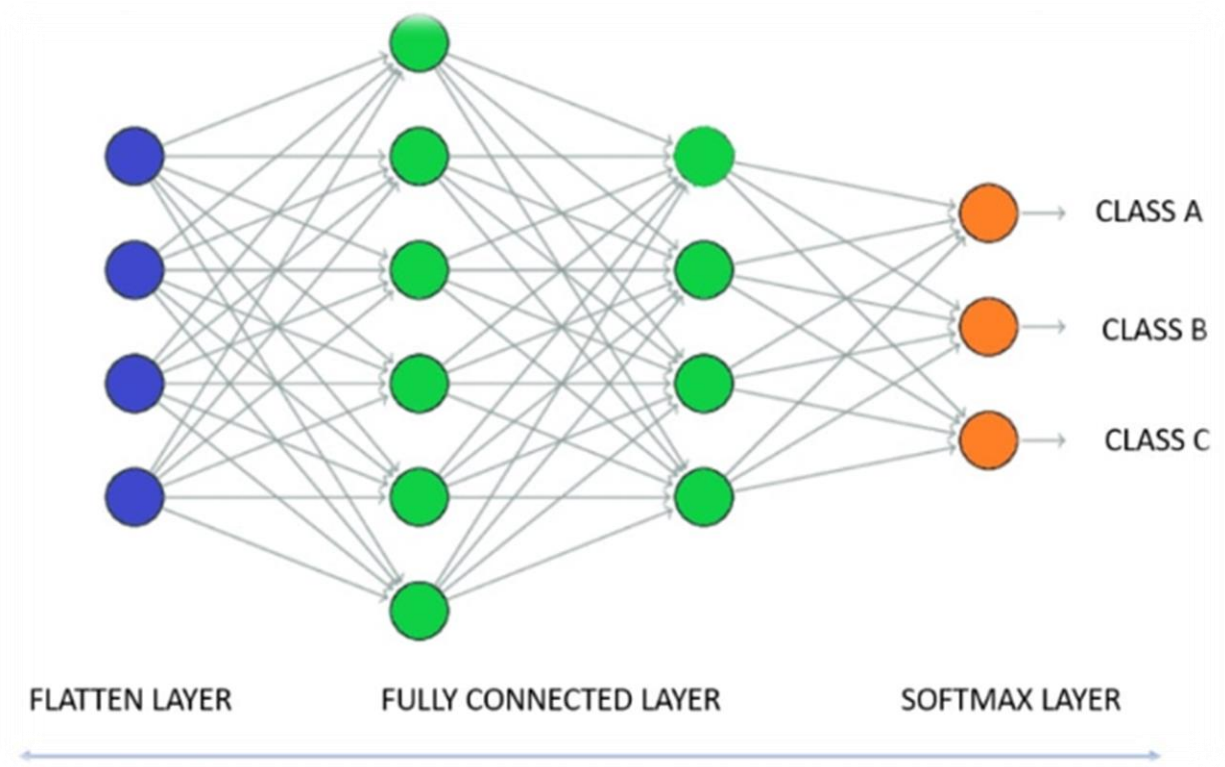
لایهٔ خروجی تمام متصل: احتمالات نهایی را برای هر برچسب نشان می دهد.

لایه کاملاً متصل از Flatten Layer که یک لایهٔ یک بعدی است ورودی می گیرد. داده های بدست آمده از Flatten Layer به تابع غیر خطی که پیش تر گفتیم ارسال می شود.
ما می‌توانیم چندین لایه از این قبیل را بر اساس عمقی که می‌خواهیم مدل طبقه‌بندی خود را به آن ببریم، اضافه کنیم. خروجی از لایه پنهان نهایی به تابع Softmax یا Sigmoid برای توزیع احتمال بر روی مجموعه نهایی تعداد کل کلاس ها ارسال می شود. کلاسی که بیشترین احتمال را دارد کلاس نهایی تصویر ورودی است.[۱۱]

به عبارت ساده تر، لایه کاملاً متصل تمام اطلاعات جمع آوری شده توسط CNN تا کنون را می گیرد و از آن برای تصمیم گیری نهایی استفاده می کند. مثل آخرین مرحله قبل از اینکه شبکه بگوید، "بله، من تقریباً مطمئن هستم که این همان چیزی است که در تصویر وجود دارد!" این بخشی است که همه چیز را به هم گره می زند و به شبکه کمک می کند تا چیزهای پیچیده را بفهمد، درست مانند زمانی که همه قطعات یک داستان را کنار هم قرار می دهیم تا کل طرح را بفهمیم.

---

[۱۸] Fully connected
[۱۹] classifier
[۲۰] flatten



شکل ۷-۳ : نمونه ای از لایهٔ تمام متصل

۳-۴ آموزش مدل

در آموزش شبکه عصبی چندین مرحله را طی می کنیم تا شبکه نحوه طبقه بندی صحیح تصاویر را بیاموزد. این مراحل چندین بار تکرار می شوند تا دقت شبکه بهبود یابد.

مقداردهی اولیه وزن ها و فیلترها [۲۱] :

در ابتدا به طور تصادفی وزن ها و فیلترها را به شبکه اختصاص می دهیم. این فیلترها مانند قالب هایی هستند که به شبکه کمک می کنند تا ویژگی های مختلف تصاویر را تشخیص دهد.

[۲۱] Initializing Weights and Filters

انتشار رو به جلو [۲۲] :

ما تصاویر آموزشی را وارد شبکه می کنیم. شبکه این تصاویر را از طریق لایه های مختلف مانند کانولوشن، فعال سازی ReLU و pooling، پردازش می کند. این به شبکه کمک می کند تا در مورد آنچه در تصویر است پیش بینی کند.

محاسبه خطا :

پیش بینی های انجام شده توسط شبکه را با برچسب های واقعی تصاویر مقایسه می کنیم. تفاوت بین این پیش بینی ها و برچسب های واقعی خطا[۲۳] نامیده می شود. کل خطا را با جمع کردن خطاهای همه تصاویر محاسبه می کنیم.

پس انتشار [۲۴] :

سپس شبکه مشخص می کند که هر وزن و فیلتر چقدر در خطای کلی نقش داشته است. این وزن ها و فیلترها را به گونه ای تنظیم می کند که خطا را کاهش دهد. این فرآیند تنظیم پارامترها را پس انتشار می نامند.

دوره ها[۲۵] :

ما این مراحل را برای تمام تصاویر موجود در مجموعه داده خود تکرار می کنیم. یک گذر کامل از تمام تصاویر، دوره نامیده می شود. ما معمولاً چندین دوره را انجام می دهیم تا به شبکه فرصت بیشتری برای یادگیری از داده ها بدهیم. اگر دوره های زیادی را انجام دهیم، شبکه ممکن است داده ها را بیش از حد برازش[۲۷،۲۶] کند.

دقت محاسبه [۲۸] :

برای اندازه گیری میزان عملکرد شبکه، دقت را محاسبه می کنیم. این تعداد پیش بینی های صحیح تقسیم بر تعداد کل پیش بینی ها است که درصدی از تعداد تصاویری که شبکه درست کرده است را به ما می دهد.

---

[۲۲] Forward Propagation
[۲۳] loss
[۲۴] Backpropagation
[۲۵] Epochs
[۲۶] داده های آموزشی را خیلی خوب یاد می گیرد تا جایی که شروع به عملکرد ضعیف در داده های جدید و دیده نشده می کند.
[۲۷] overfit
[۲۸] Calculating Accuracy

اندازه دسته ای [۲۹] :

به جای پردازش کل مجموعه در هر تکرار، که می توان از نظر محاسباتی گران و حافظه فشرده باشد، مجموعه داده به گروه های کوچکتر تقسیم می شود. اندازه دسته تعیین می کند که چه تعداد نمونه در هر دسته گنجانده شده است.

[۲۹] Batch Size

فصل ۴

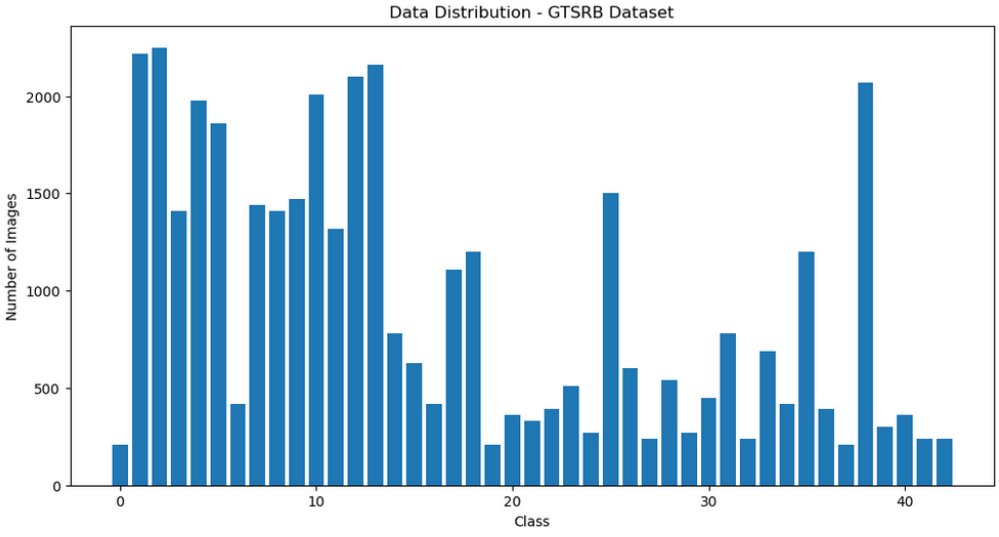
پیاده سازی

حال در این فصل اقدام به پیاده سازی پروژه ی تشخیص خودکار علائم راهنمایی و رانندگی به کمک یادگیری عمیق به زبان پایتون میکنیم. در این پروژه ما از پایگاه داده ی GTSRB (German Traffic Sign Benchmark)، که علائم ترافیکی کشور آلمان می باشد برای اموزش و ازمایش داده ها استفاده کردیم.

این پایگاه داده شامل حدود سی و پنج هزار عکس می باشد که درون چهل و سه کلاس مختلف قرار گرفته اند و هر کلاس مربوط به یک علائم راهنمایی و رانندگی می باشد.

تصاویر پایگاه داده، دارای سایز ۳۲ در ۳۲ پیکسل و به صورت سه کاناله (همان RGB) می باشند که در فصل های پیش درباره ی این نوع عکس ها گفتیم.

همچنین همانطور که از شکل زیر مشاهده شد، توزیع تصاویر درون ۴۳ کلاس به صورت برابر نمی باشد. یعنی تعداد تصاویر درون هر کدام از کلاس ها متفاوت از دیگری می باشد. با این حال این عدم توازن تاثیری در دقت تشخیص علائم نمی گذارد.



شکل ۱-۴ : توزیع داده ها

همانطور که پیش تر و در فصل پیش پرازش به عملیات های پیش پردازش پرداختیم، حال سه عملیات تبدیل تصویر از RGB به خاکستری، نرمالسازی تصاویر و یکسان سازی هیستوگرام را پس از خواندن تصاویر آموزشی از پوشه ی مورد نظر در پایتون اعمال میکنیم.

```python
# Load the images and labels from the dataset
images = []
labels = []

for class_folder in range(43):

    # Convert the class_folder to a 5-digit zero-padded string
    class_folder_str = str(class_folder).zfill(5)

    class_path = os.path.join(dataset_folder, class_folder_str)

    for image_file in os.listdir(class_path):
        image_path = os.path.join(class_path, image_file)

        image = imageio.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # Convert RGB image to grayscale
        image = exposure.equalize_hist(image) # Perform histogram equalization
        image = image / 255.0 # Normalize the image by dividing by 255.0
        image = cv2.resize(image, (32, 32)) # Resize images to a consistent size

        images.append(image)
        labels.append(class_folder)
```

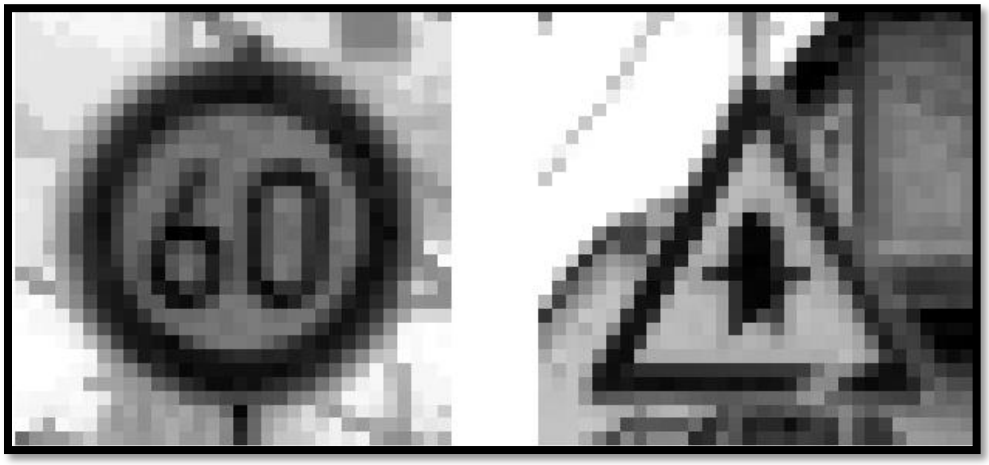
شکل ۲-۴ : عملیات پیش پردازش مدل در پایتون

در گام بعد مجموعه ی شامل تصاویر و برچسب های خود را در قالب آرایه های NumPy نگهداری می کنیم که این تبدیل می‌تواند مفید باشد زیرا این امکان را می‌دهد عملیات‌های عددی، محاسبات و دستکاری‌های مختلف را روی داده‌ها با کارآمدتر و راحت‌تر از استفاده از لیست‌های ساده پایتون یا سایر ساختارهای داده انجام دهیم.

```python
# Convert the images and labels to NumPy arrays
images = np.array(images)
labels = np.array(labels)
```

شکل ۳-۴ : تبدیل مجموعه ی شامل تصاویر و برچسب های خود به قالب آرایه های **NumPy**

پس از اعمال عملیات بالا بر روی تصاویر پایگاه داده خود، دو تصویر را از پایگاه داده به صورت تصادفی فراخواندیم که تصاویری مانند شکل زیر حاصل شد :



شکل ۳-۴ : تصاویر نمونه بعد از اعمال سه عملیات پیش پردازش

در مرحله ی بعد افزایش داده را روی تصاویر آموزشی خود اعمال کردیم که پیش تر و در فصل دوم این مراحل را نشان دادیم.(برای اطلاعات بیشتر به صفحه شماره ۱۹ مراجعه کنید.)

همانطور که در فصل سوم درباره ی مدلسازی شبکه ی عصبی کانولوشنی گفتیم، حال در ادامه در زبان پایتون مراحل ان را پیاده سازی می کنیم.

تصویر ورودی ما همانطور که گفتیم دارای مقیاس خاکستری و شامل ۳۲ در ۳۲ پیکسل می باشد.

لایه کانولوشن اولیه شامل دو مجموعه ۶۰ فیلتری است که هر کدام با اندازه ۵ در ۵ پیکسل هستند. این فرآیند به برجسته کردن ویژگی های مهم در تصاویر کمک می کند.

سپس MaxPooling برای کوچک کردن داده‌ها با انتخاب حداکثر مقادیر پیکسل استفاده می‌شود.

فرآیند کانولوشن، همراه با تابع فعال‌سازی ReLU، بر ویژگی‌های مهم تأکید می‌کند.

این مدل از یک لایه کانولوشنال دوم با ۳۰ فیلتر به اندازه ۳ در ۳ استفاده می‌کند که به دنبال آن فعال‌سازی ReLU و یک لایه MaxPooling دیگر انجام می‌شود.

برای جلوگیری از برازش بیش از حد، پس از مجموعه دوم کانولوشن، از dropout[۳۰] استفاده می‌شود. داده‌های حاصل از لایه‌های کانولوشن، قبل از وارد شدن به لایه‌های تمام متصل [۳۱]برای تشکیل یک بردار مسطح[۳۲] می‌شوند.

مراحل پایانی مدل از دو لایه تمام متصل تشکیل شده است. اولین مورد از فعال‌سازی ReLU استفاده می‌کند و لایه دوم دارای اندازه ۴۳ است که نشان دهنده تعداد کلاس های علائم راهنمایی و رانندگی است.

تابع SoftMax به عنوان تابع فعال سازی نهایی استفاده می شود، بردار خروجی را عادی می کند و یک تفسیر احتمالی را تسهیل می کند، که به پیش بینی محتمل ترین کلاس علائم راهنمایی کمک می کند.

پیاده سازی این مدل در پایتون به شرح زیر است :

```
# Define the CNN model
model = Sequential()

model.add(Conv2D(60, kernel_size=(5, 5), activation='relu', input_shape=(32, 32, 1)))
model.add(Conv2D(60, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(30, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(30, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))
model.add(Flatten())

model.add(Dense(500, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(43, activation='softmax'))
```

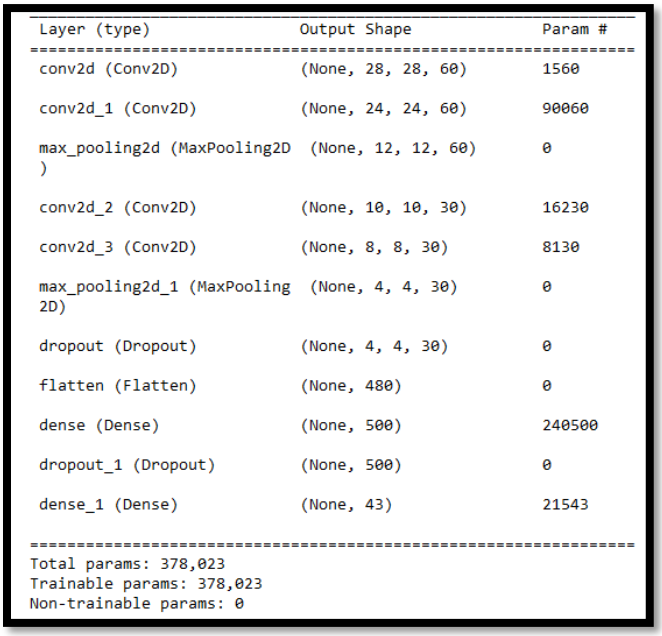
شکل ۴-۴ : پیاده سازی CNN مدل در پایتون

[۳۰] به طور تصادفی برخی از نورون ها را با احتمال مشخصی غیرفعال می کند. این بدان معناست که آن نورون‌ها برای مدت کوتاهی به فرآیند یادگیری کمک نمی‌کنند.و بدین ترتیب نورون‌های دیگر را تشویق می‌کند تا قدم بردارند و یاد بگیرند که از عهده کار برآیند، و از تخصصی شدن بیش از حد هر نورون منفردی جلوگیری می‌کند.

[۳۱] Fully connected(or dense)

[۳۲] flatten

در ادامه به کمک دستور model.summary() در پایتون، میتوان خلاصه ای عملیات خود را دید :



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 60)        1560

conv2d_1 (Conv2D)            (None, 24, 24, 60)        90060

max_pooling2d (MaxPooling2D  (None, 12, 12, 60)        0
)

conv2d_2 (Conv2D)            (None, 10, 10, 30)        16230

conv2d_3 (Conv2D)            (None, 8, 8, 30)          8130

max_pooling2d_1 (MaxPooling  (None, 4, 4, 30)          0
2D)

dropout (Dropout)            (None, 4, 4, 30)          0

flatten (Flatten)            (None, 480)               0

dense (Dense)                (None, 500)               240500

dropout_1 (Dropout)          (None, 500)               0

dense_1 (Dense)              (None, 43)                21543

=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
```

شکل ۵-۴ : خلاصه ای از معماری مدل

در ادامه باید مدل خود را کامپایل کنیم یعنی نحوه یادگیری آن از داده ها را پیکربندی کنیم.

بدین منظور از پارامتر loss که پیش تر تعریف کردیم استفاده می کنیم و در این مورد « categorical_crossentropy» انتخاب می شود که معمولا برای مسائل طبقه بندی چند کلاسه است.

از پارامتر بهینه ساز[۳۳]، برای به روزرسانی وزن مدل و به حداقل رسوندن تابع loss استفاده می کنیم و در این مورد «adam» انتخاب می شود که یک الگوریتم بهینه سازی است که به دلیل کارایی و اثربخشی ان، در بسیاری از سناریو ها مورد استفاده قرار می گیرد.

پارامتر اخر، «metrics» است که لیستی از معیارهای ارزیابی است که برای نظارت بر عملکرد مدل در طول آموزش استفاده می شود. «دقت[۳۴]» یک معیار رایج برای کارهای طبقه‌بندی است که درصد نمونه‌های طبقه‌بندی صحیح را نشان می‌دهد.

[۳۳] optimizer
[۳۴] accuracy

نحوه کامپایل مدل در پایتون به شرح زیر می باشد:

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

شکل ۶-۴ : کامپایل مدل

در ادامه برای اینکه پردازش و یادگیری از داده ها برای مدل یادگیری ماشین اسان تر باشد، برچسب های طبقه بندی شده را به بردار های رمزگذاری شده تبدیل می کنیم که یک مرحله ی رایج در یادگیری ماشین می باشد.

در رمزگذاری one-hot هر برچسب را به صورت یک بردار باینری نشان می دهدکه در ان هر موقعیت مربوط به یک کلاس است و موقعیت کلاسی که نمونه به آن تعلق دارد با یک مشخص می شود و موقعیت های دیگر با صفر پر می شود.

و در پایتون به صورت زیر اعمال می شود:

```
# Convert labels to one-hot encoded vectors
num_classes = 43
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
```

شکل ۷-۴ : تبدیل برچسب های طبقه بندی شده به بردارهای رمزگذاری شده

حال زمان آموزش مدل است. بدین منظور از پارامتر های epoch و bach_size استفاده کردیم که همانطور که پیش تر در فصل قبل گفتیم epoch، تعداد دفعاتی که مدل از کل مجموعه داده آموزشی در طول آموزش عبور می کند را تعیین می کند و bach_size، تعداد نقاط داده پردازش شده قبل از به‌روزرسانی مدل. پارامتر دیگر داده های اعتبار سنجی است که مانند آزمون کوچک برای ماشین در طول یک دوره آن است. اعتبارسنجی برای نظارت بر عملکرد مدل در داده‌های دیده نشده و جلوگیری از برازش بیش از حد استفاده می‌شود.

برای فرایند آموزش در پایتون از تابع model.fit() استفاده می کنیم که مدل یادگیری ماشین را با استفاده از داده های آموزشی ارائه شده آموزش می دهد.

در ابتدا مدل را با epoch = ۱۵ و bach_size = ٦٤ آموزش دادیم که دقتی برابر ۹۳ درصد بدست آمد.

```
Epoch 15/15
491/491 [==============================] - 204s 415ms/step - loss: 0.7695 - accuracy: 0.7577 - val_loss: 0.2375 - val_accu
racy: 0.9338
```

شکل۸-۴ : دقت مدل با دوره ی ۳۰

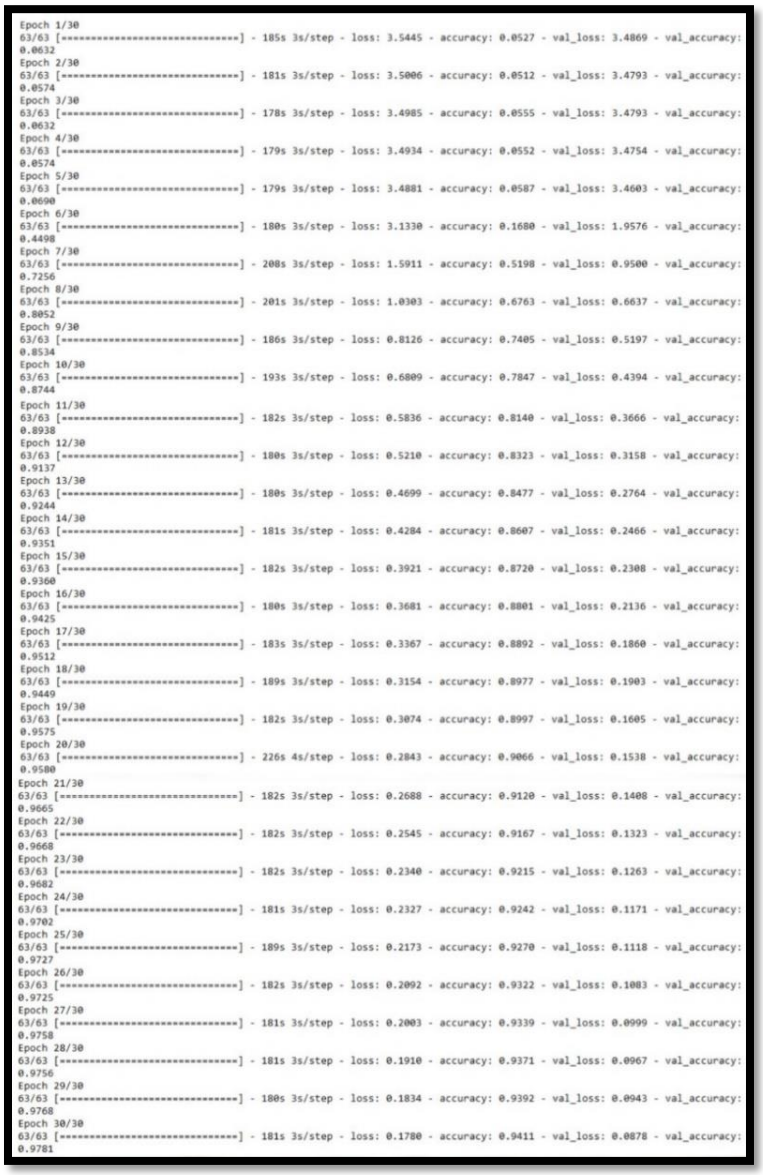
نتیجه نشان داد که همچنان قابلیت اموزش بهتر را دارد بنابراین اینبار مدل را با epoch را برابر با ۳۰ و bach_size برابر با ۵۰۰ آموزش دادیم که دقتی برابر با ۹۷.۸۱ درصد بدست آمد.

```
# Train the model

history = model.fit(X_train, y_train, batch_size=500,
                    epochs=30,
                    validation_data=(X_test, y_test) )
```

شکل ۹-۴ : اموزش مدل در پایتون

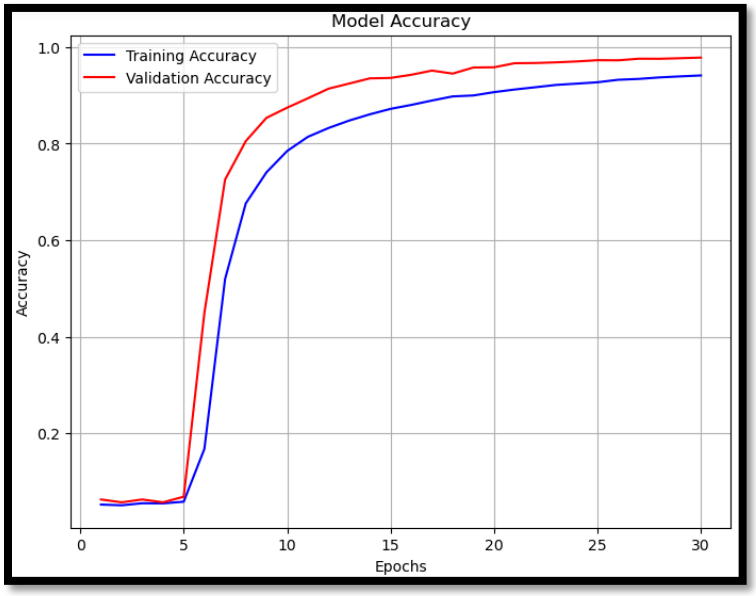
خروجی حاصل از آموزش نیز به شرح زیر شد:



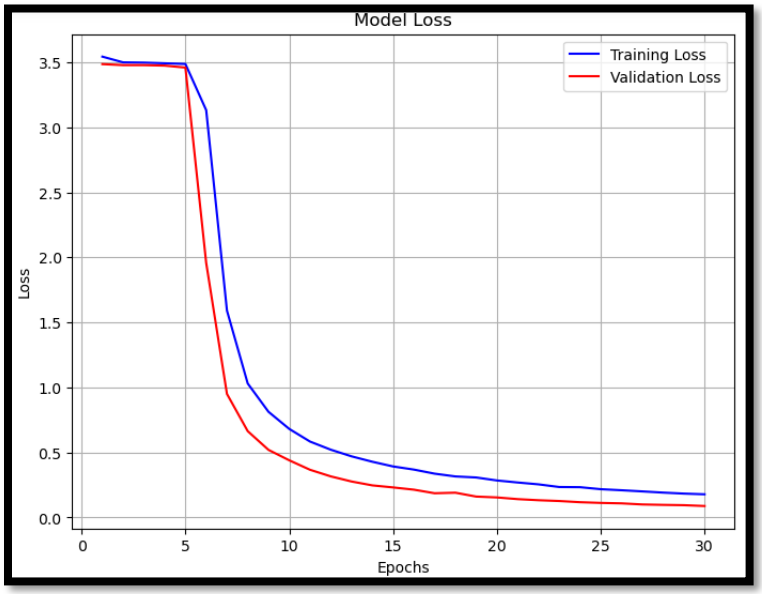
شکل ۱۰-۴ : فرایند اموزش مدل با دقت ٪۹۷.۸۱

همانطور که مشخص است توانستیم با بالا بردن epoch از ۱۵ به ۳۰ و بالا بردن bach_size از ۶۴ به ۵۰۰ به دقتی برابر ۹۷ درصد برسیم.

سپس با دیدن نمودار های زیر مشخص کردیم که مدل یادگیری ما چقدر در تکرار های مختلف خوب اموزش می بیند.



شکل ۱۱-۴ : نمودار دقت مدل در تشخیص اشیا



شکل ۱۲-۴ : نمودار خطای مدل

افزایش دقت آموزشی نشان می‌دهد که مدل در حال یادگیری و بهبود پیش‌بینی‌های خود بر روی داده‌های آموزشی است. این نشان دهنده پیشرفت در آموزش است و نشان می دهد که مدل الگوها و روابط مربوطه را در داده ها ثبت می کند.

کاهش loss در طول تمرین نشان می دهد که مدل در طول زمان پیش بینی های خود را بهبود می بخشد و به سمت دقت بهتر همگرا می شود.

در ادامه برای تست مدل خود، پس از مشخص کردن پوشه ی شامل تصاویر تست خود، نام ۴۳ کلاس خود را مشخص کردیم(شامل محدودیت سرعت ۲۰ کیلومتر بر ساعت، محدودیت سرعت ۳۰ کیلومتر در ساعت، رعایت حق تقدم و ....)

پس از این موارد برای تست مجموعه ۴ تصاویر را به صورت تصادفی از پوشه ی تصاویر آزمایشی خود انتخاب کرده و با نمایش تصاویر مشخص می کنیم که متعلق به کدام علائم راهنمایی و رانندگی می باشد.

پس از گرفتن ۴ تصویر تصادفی، کاری که در مرحله ی پیش پردازش خود انجام دادیم را روی این تصاویر نیز اعمال می کنیم (یعنی همان تبدیل به خاکستری و یکسان سازی هیستوگرام و نرمالسازی و تبدیل به سایز ۳۲ در۳۲).

سپس تصاویر ازمایشی خود را از مدل اموزش دیده خود عبور می دهیم تا احتمالات پیش بینی شده ی هر کلاس را برای هر تصویر پیش بینی کند که این کار در پایتون به صورت زیر انجام می شود:

```
# Make predictions using the trained model
predictions = model.predict(test_images)
```

```
1/1 [==============================] - 0s 69ms/step
```

شکل ۱۳-۴ : محاسبه ی احتمال پیش بینی شده برای هر تصویر

در ادامه میخواهیم برای هر ۴ تصویری که به صورت تصادفی از مجموعه ی داده ی آزمایشی خود انتخاب کردیم، دوتا از کلاس هایی که با احتمال بیشتری تصویر به ان تعلق دارد به همراه احتمالشان نشان داده شود.

```
# get the top two predicted class indices and their corresponding probabilities
top_class_indices = np.argsort(predictions, axis=1)[:, -2:]
top_class_probs = np.sort(predictions, axis=1)[:, -2:]
```

شکل ۱۴-۴ : شاخص ها و احتمالات دو کلاس پیش بینی شده برتر برای هر تصویر در پایتون

حال شاخص های دو کلاس پیش بینی شده برتر برای هر تصویر را به نام کلاس مربوطه تبدیل می کنیم.

```
# Map the predicted class numbers to class names for the top tow classes
top_class_names = [[class_names[idx] for idx in indices] for indices in top_class_indices]
```

شکل ۱۵-۴ : کد تبدیل شاخص های دو کلاس برتر به نام کلاس مربوطه

سپس دو کلاس با احتمال تعلق بیشتر همراه با احتمالشان برای هر کدام از ۴ تصاویر به شکل زیر محاسبه شد:

```
# display the names and probabilities of the two most probable classes for each image
for i in range(num_images_to_test):
    print(f"Image: {test_image_names[i]}")
    for j in range(2):
        print(f"Top Class {j + 1}: {top_class_names[i][j]}, Probability: {top_class_probs[i][j]:.2f}")
    print("-" * 40)
```

```
Image: 07437.ppm
Top Class 1: Speed limit (30km/h), Probability: 0.47
Top Class 2: Speed limit (50km/h), Probability: 0.52
----------------------------------------
Image: 02578.ppm
Top Class 1: Traffic signals, Probability: 0.00
Top Class 2: Bumpy road, Probability: 1.00
----------------------------------------
Image: 00170.ppm
Top Class 1: No vehicles, Probability: 0.00
Top Class 2: Yield, Probability: 1.00
----------------------------------------
Image: 03111.ppm
Top Class 1: Wild animals crossing, Probability: 0.00
Top Class 2: Road work, Probability: 1.00
----------------------------------------
```

شکل ۱۶-۴ : نمایش دوتا کلاس با بیشترین احتمال برای هر تصویر

همانطور که از نتیجه پیداست به طور مثال برای تصویر اول با احتمال ۴۷ درصد تصویر نشان دهنده ی محدودیت سرعت ۳۰ کیلومتر بر ساعت و با احتمال ۵۲ درصد نشان دهنده ی محدودیت سرعت ۵۰ کیلومتر بر ساعت می باشد. بنابراین انتظار می رود تصویر اول بیانگر محدودیت سرعت ۵۰ کیلومتر بر ساعت باشد.

در ادامه چهار تصاویر پیشنهادی را که به صورت رندوم انتخاب شده بود باید با نام کلاس پیش بینی شده نمایش دهیم.

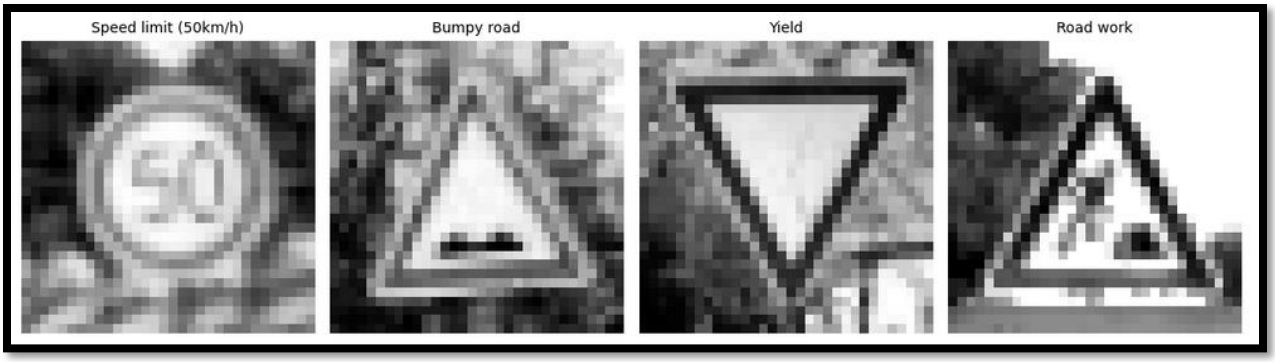
```
# Visualize the test images along with their predicted Class names
plt.figure(figsize=(12, 6))
for i in range(num_images_to_test):
    plt.subplot(1, num_images_to_test, i + 1)
    plt.imshow(test_images[i].squeeze(), cmap='gray')
    if top_class_probs[i][0] > top_class_probs[i][1]:
        plt.title(top_class_names[i][0] , fontsize=10)
    else:
        plt.title(top_class_names[i][1] , fontsize=10)
    plt.axis('off')

plt.tight_layout()
plt.show()
```

شکل ۱۷-۴ : کد پایتون برای نمایش تصاویر ازمایشی همراه کلاس پیش بینی شده

در نهایت با توجه به کد بالا چهار تصویر ازمایشی خود با کلاس پیش بینی شده ی ان نمایش داده شد.

و با توجه به تصاویر و کلاس پیش بینی شده نشان می دهد که مدل به درستی عمل کرده است و توانسته است به خوبی علائم راهنمایی را تشخیص دهد.



شکل ۱۸-۴ : نمونه تصاویر آزمایش شده

همانطور که از شکل پیداست، تصویر اول مربوط به محدودیت سرعت ۵۰ کیلومتر بر ساعت می باشد و در خروجی نیز نام کلاس پیش بینی شده همین مسئله را نشان می دهد و همانطور که پیش تر نشان داده بودیم احتمال تعلق این تصویر به این کلاس ۵۲ درصد پیش بینی شده بود که بیشترین احتمال را نیز شامل بود.

بقیه کلاس ها نیز به همین ترتیب و به درستی تشخیص داده شده است.

کد پروژه در گیت هاب قابل مشاهده است.

فصل ۵

نتیجه گیری

در این پروژه، ما شروع به توسعه یک مدل یادگیری عمیق برای تشخیص علائم ترافیکی با استفاده از مجموعه داده‌های معیار تشخیص علائم ترافیک آلمان (GTSRB) کردیم. هدف ساخت مدلی بود که بتواند علائم راهنمایی و رانندگی مختلفی را که معمولاً در جاده ها با آن مواجه می شوند، به طور دقیق طبقه بندی کند. در طول پروژه، ما از یک رویکرد سیستماتیک پیروی کردیم که شامل بارگذاری داده، پیش پردازش، طراحی مدل، آموزش و ارزیابی بود. ما یک شبکه عصبی کانولوشن (CNN) را به عنوان انتخاب خود برای وظایف بینایی رایانه انتخاب کردیم. این تصمیم به دلیل عملکرد برتر و سهولت پیاده سازی آن در مقایسه با سایر شبکه ها بود. CNN ها از نظر دقت عالی هستند. آنها با شناسایی مستقل ویژگی ها بدون دخالت انسان از پیشینیان خود بهتر عمل می کنند.

در نتیجه، مدل یادگیری عمیق ما با موفقیت به چالش تشخیص علائم راهنمایی و رانندگی پرداخت. با استفاده از معماری خوب طراحی شده CNN، پیش پردازش موثر داده ها، و تکنیک های تقویت، مدلی را ایجاد کردیم که دقت و توانایی های تعمیم قابل توجهی را به نمایش گذاشت. این پروژه بر قدرت یادگیری عمیق در حل وظایف طبقه بندی تصاویر در دنیای واقعی تاکید می کند و پتانسیل آن را برای افزایش ایمنی جاده از طریق سیستم های تشخیص خودکار علائم ترافیکی برجسته می کند.

در سناریوی خود، ما تصاویر متعددی از علائم ترافیکی را به شبکه دادیم و آن را قادر می‌سازیم تا ویژگی‌های منحصربه‌فردی را برای هر دسته علائم بیاموزد. این رویکرد نرخ دقت قابل توجه ۹۷ درصد را به همراه داشت.

مراجع

[١] **https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained**

[٢] **https://www.wgu.edu/blog/what-convolutional-neural-network٢٠٠٨.html#close**

[٣] **https://www.baeldung.com/cs/convert-rgb-to-grayscale**

[٤] **https://www.analyticsvidhya.com/blog/٢٠٢٢/٠١/histogram-equalization/**

[٥] **https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-٨-why-should-we-normalize-image-pixel-values-or-divide-by-٢٥٥-٤٦٠٨ac٥cd٢٦a**

[٦] https://towardsdatascience.com/image-data-augmentation-for-deep-learning-٧٧a٨٧fabd٢bf#:~:text=Image%٢٠data%٢٠augmentation%٢٠is%٢٠a,the%٢٠image%٢٠horizontally%٢٠or%٢٠vertically

[٧] **https://towardsdatascience.com/convolutional-neural-networks-explained-٩cc٥١٨٨c٤٩٣٩**

[٨] **https://www.v٧labs.com/blog/neural-networks-activation-functions**

[٩] **https://towardsdatascience.com/convolutional-neural-networks-explained-٩cc٥١٨٨c٤٩٣٩**

[١٠] **https://towardsdatascience.com/convolutional-neural-networks-explained-٩cc٥١٨٨c٤٩٣٩**

[١١] https://indiantechwarrior.com/generative-adversarial-networks/