
**دانشگاه صنعتی نوشیروانی بابل**

دانشکده برق و کامپیوتر

گزارش پروژه پیاده‌سازیCNN بینایی رایانه

# تشخیص چهره در نورپردازی‌های گوناگون

**نگارش:**

جمال الدین دمیرچی: ۹۸۳۲۱۲۰۹۳

**استاد درس:**

دکتر ازوجی

زمستان ۱۴۰۲


بسم الله الرحمن الرحیم
MRTsoft

## فهرست مطالب

**عنوان** **شماره صفحه**

## فهرست شکل‌ها

## فهرست جدول‌ها

**عنوان** **شماره صفحه**

# چکیده گزارش

در این پروژه، یک مدل شبکه عصبی کانولوشنال (CNN) برای دسته‌بندی تصاویر افراد طراحی و پیاده‌سازی شد. این پروژه با هدف افزایش دقت در تشخیص و دسته‌بندی تصاویر انجام شده است. مراحل اصلی پروژه شامل پیش‌پردازش داده، تعریف معماری مدل، آموزش، ارزیابی و نمایش نتایج می‌شوند.

در مرحله پیش‌پردازش، داده‌ها از دایرکتوری مربوطه خوانده شده و به دسته‌بندی‌های مختلف منتقل شده‌اند. سپس مدل CNN با استفاده از لایه‌های Convolutional طراحی شده و با تنظیم پارامترهای هایپر، مانند تعداد ایپاک و نرخ یادگیری، آموزش داده شده است.

نتایج نشان داده‌اند که مدل با دقت و بازخوانی بسیار خوبی توانسته تصاویر را به درستی دسته‌بندی کند. از این رو، مدل طراحی شده برای دسته‌بندی تصاویر افراد به عنوان یک روش موثر و قابل قبول ثابت شده است.

## مقدمه

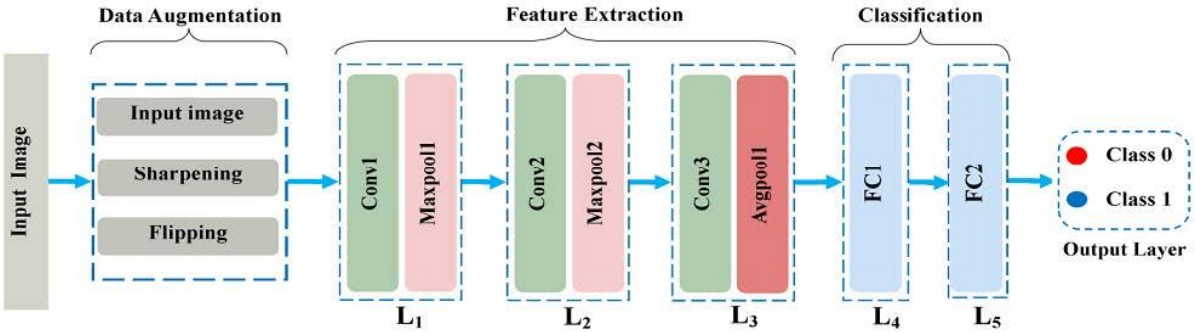
در این پروژه، به تحقیق و تحلیل تصاویر تمام رخ و هم‌تراز شده با ابعاد ۱۶۰ × ۱۶۰ پیکسل از ۱۰ شخص مختلف پرداخته‌ایم. این تصاویر تحت ۶۴ نورپردازی گوناگون از هر فرد به دست آمده‌اند، و این نورپردازی‌ها با توجه به اندازه و زاویه آن‌ها، به ۵ دسته گوناگون دسته‌بندی شده‌اند.

برای دستیابی به دقت و کارایی بالاتر در دسته‌بندی تصاویر، از یک شبکه عصبی کانولوشنی به عنوان معماری اصلی استفاده می‌شود.

برای ارزیابی کیفیت و عملکرد شبکه، در طی ۱۰۰ ایپاک از آن استفاده شده و نمودار Loss شبکه در طول این دوره‌ها ثبت شده است. این نمودار اطلاعات مهمی ارائه می‌دهد که به تحلیل پیشرفت آموزش شبکه کمک می‌کند.

همچنین، دقت دسته‌بندی و امتیاز F1 Score بر روی مجموعه‌داده آزمایشی گزارش شده‌اند. این اطلاعات نشان دهنده کارایی عملیات دسته‌بندی شبکه در تشخیص و دسته‌بندی تصاویر با توجه به معیارهای متفاوت ارزیابی می‌شوند.

## معماری شبکهCNN (CNN network architecture) :




شکل۱) معماری شبکه CNN

### Data Augmentation:

افزایش داده یا "Data Augmentation" از این تکنیک استفاده می‌شود تا به صورت مصنوعی اندازه و تنوع مجموعه داده آموزشی را با ایجاد نسخه‌های تغییر یافته از داده‌های موجود افزایش دهد. این عمل کمک می‌کند تا عمومی‌سازی مدل و جلوگیری از بیش‌برازش (overfitting) ایجاد شود.

در این معماری از روش‌های Sharpening و Flipping بدین منظور استفاده شده است.

### Feature Extraction:

استخراج ویژگی، فرآیند شناسایی و استخراج ویژگی‌های مرتبط از داده است. سپس از این ویژگی‌ها برای آموزش مدل طبقه‌بندی استفاده می‌شود.

در زمینه تشخیص تصاویر، ویژگی‌ها ممکن است شامل لبه‌ها، شکل‌ها، بافت‌ها و رنگ‌ها باشند. این ویژگی‌ها با استفاده از روش‌های مختلفی استخراج می‌شوند، در این پروژه از CNN استفاده شده است:

شبکه‌های عصبی کانولوشنی: CNN ها به طور خاص برای استخراج ویژگی‌ها از تصاویر طراحی شده‌اند. این شبکه‌ها با اعمال یک سری فیلتر به تصویر کار می‌کنند که یاد می‌گیرند الگوهای خاصی را شناسایی کنند.در تصویر و همچنین در ادامه فیلترهای استفاده شده بررسی می‌شود.

### Average Pooling:

میانگین مقدار تمام عناصر در یک منطقه مشخص از نقشه ویژگی را محاسبه می‌کند. این منطقه مانند یک فیلتر عمل می‌کند که در طول نقشه حرکت کرده و برای هر موقعیت یک مقدار واحد را خروجی می‌دهد.

### Max Pooling:

مقدار بیشینه در یک منطقه مشخص از نقشه ویژگی را انتخاب می‌کند. فیلتر دوباره در طول نقشه حرکت کرده و برای هر منطقه، فعال‌ترین فیچر را حفظ می‌کند.

هردو باعث می‌شوند تا اندازه نقشه را کاهش داده و ویژگی‌های چشم‌گیری مانند لبه‌ها و گوشه‌ها را تأکید می‌دهد. این باعث می‌شود که شبکه در مقابل ترجیحات کوچک و انحراف‌های ورودی مقاوم‌تر شود.
موارد استفاده: محبوب برای وظایفی مانند طبقه‌بندی تصاویر که شناسایی ویژگی‌های کلیدی حیاتی است. این به شبکه کمک می‌کند تا روی ویژگی‌های برجسته‌تر یک شیء تمرکز کند و جزئیات غیرضروری را نادیده بگیرد.

## Classification:

طبقه‌بندی فرآیند پیش‌بینی برچسب کلاس یک نقطه داده جدید است.
معمولاً مدل‌های طبقه‌بندی با استفاده از یادگیری نظارت‌شده آموزش می‌بینند، که به معنای آن است که به آن‌ها یک مجموعه از داده‌های دارای برچسب (تصاویر با برچسب‌های کلاس متناظر) داده می‌شود. مدل یاد می‌گیرد که ویژگی‌های یک تصویر را به برچسب کلاس مرتبطش نگاشت دهد.

در ادامه جزئیات معماری پیشنهادی CNN که توضیح داده شد در جدول ۱ آورده شده است:

| Layer Name | Layer Type | No. of Filters | Kernel Size | Stride | Input Features | Output Features |
|---|---|---|---|---|---|---|
| L1 | Convolution (Conv1) | 16 | (3x3) | (1x1) | (1,160,160) | (16,158,158) |
| | Batch Normalization | - | - | - | (16,218,108) | (16,218,108) |
| | ReLU | - | - | - | (16,218,108) | (16,218,108) |
| | Max Pooling (Maxpool1) | - | (4x4) | (4x4) | (16,218,108) | (16,54,27) |
| L2 | Convolution (Conv2) | 32 | (3x3) | (1x1) | (16,54,27) | (32,52,25) |
| | Batch Normalization | - | - | - | (32,52,25) | (32,52,25) |
| | ReLU | - | - | - | (32,52,25) | (32,52,25) |
| | Max Pooling (Maxpool2) | - | (2x2) | (2x2) | (32,52,25) | (32,26,12) |
| L3 | Convolution (Conv3) | 64 | (3x3) | (1x1) | (32,26,12) | (64,24,10) |
| | Batch Normalization | - | - | - | (64,24,10) | (64,24,10) |
| | ReLU | - | - | - | (64,24,10) | (64,24,10) |
| | Average Pooling (Avgpool1) | - | (2x2) | (2x2) | (64,24,10) | (64,2,2) |
| L4 | Fully Connected (FC1) | - | - | - | 256 | 256 |
| | ReLU | - | - | - | 256 | 256 |
| L5 | Fully Connected (FC2) | - | - | - | 256 | 148 |

جدول ۱) جزئیات معماری CNN پیشنهادی در شکل ۱

**ابرپارامترها(Hyperparameter):**

در این قسمت Hyperparameterهای استفاده شده برای تنظیم CNN معرفی می‌شوند.

| Hyperparameter | Value |
|---|---|
| Batch size | 8 |
| Number of epochs | 100 |
| Initial learning rate | 0.001 |
| L2 regularization | 0.001 |

جدول۲) تنظیمات پارامترهای CNN پیشنهادی در شکل۱

Batch size: تعداد نمونه‌های آموزشی است که در هر تکرار برای به‌روزرسانی وزن‌های مدل استفاده می‌شود. یک اندازه دسته کوچکتر می‌تواند به مدل کمک کند تا از داده‌های متنوع‌تر در هر به‌روزرسانی استفاده کند، اما ممکن است همچنین باعث کند شدن فرآیند همگرایی گردد. اندازه دسته بزرگتر می‌تواند همگرایی را تسریع بخشد، اما ممکن است نسبت به داده‌های نویزی حساس‌تر باشد. در اینجا از اندازه دسته ۸ استفاده شده است.

Number of epochs: تعداد دفعاتی است که مدل از سراسر مجموعه داده‌های آموزشی می‌گذرد. ایپاک‌های بیشتر به مدل این امکان را می‌دهد تا الگوهای پیچیده‌تری را یاد بگیرد، اما ایپاک‌های زیاد می‌تواند به بیش‌برازش منجر شود. در اینجا، مدل برای ۱۰۰ ایپاک آموزش دیده است.

Initial learning rate: این مقدار کنترل می‌کند چقدر وزن‌های مدل در هر تکرار به‌روزرسانی می‌شوند. یک نرخ یادگیری بالاتر می‌تواند منجر به یادگیری سریعتر شود، اما همچنین ممکن است مدل را ناپایدار و به بیش‌آموزش حساس کند. یک نرخ یادگیری پایین‌تر استقرار بیشتری دارد اما ممکن است طولانی‌تر زمان ببرد تا همگرایی یابد. در اینجا از یک نرخ یادگیری اولیه ۰.۰۰۱ استفاده شده است.

L2 regularization: یک تکنیک برای جلوگیری از بیش‌برازش است که با توجه به بزرگی وزن‌های مدل، به نحوی جریمه می‌دهد.L2 بالاتر به وزن‌های کوچکتر و مدل‌های کم‌پیچیده‌تر، که کمتر احتمال بیش‌برازش دارند، منجر می‌شود. در اینجا از L2 برابر با ۰.۰۰۱ استفاده شده است.

مقادیر ابرپارامترها به خصوصیت مجموعه داده مربوط هستند.

**تحلیل مسئله پروژه :**

برای حل مساله، ابتدا دو شخص به دلخواه را انتخاب می‌کنیم. تمام تصاویر مربوط به شخص اول با برچسب 1 و تمام تصاویر مربوط به شخص دوم با برچسب 0 مشخص می‌شوند. سپس، زیرمجموعه‌ای حاوی دو شخص به عنوان مجموعه آزمون تعیین می‌شود، و تصاویر باقیمانده این دو شخص به عنوان مجموعه آموزشی مورد استفاده قرار می‌گیرند. به این ترتیب، مواجه با یک مسئله دسته‌بندی دو کلاسه خواهیم بود.

**پیاده‌سازی:**

**کتابخانه‌های استفاده شده(Libraries) :**

در این قسمت از کد، برای پیش‌پردازش تصویر و ساخت یک مدل شبکه عصبی همگام(CNN) کتابخانه "Keras" Import می‌شود.

کتابخانه "os" برای انجام وظایف مدیریت فایل مانند دسترسی و تعامل با فایل‌ها و دایرکتوری‌ها.

ماژول"keras.preprocessing" به عنوان"preprocess" برای بارگذاری و پیش‌پردازش تصاویر. این ماژول توابع مختلفی برای پیش‌پردازش تصاویر ارائه می‌دهد، مانند تغییر اندازه، برش و نرمال‌سازی.

کلاس "Sequential" از ماژول "keras.models" برای ساخت یک مدل توالی. یک مدل توالی یک پشته خطی از لایه‌ها است، جایی که هر لایه دقیقاً یک تنسور ورودی و یک تنسور خروجی دارد.

کلاس "Conv2D" از ماژول "keras.layers" برای ایجاد یک لایه کانولوشن 2D. لایه‌های کانولوشن هسته اصلی ساختارهای CNN هستند و برای استخراج ویژگی‌ها از تصاویر ورودی استفاده می‌شوند.

کلاس "BatchNormalization" از ماژول "keras.layers" برای افزودن نرمال‌سازی دسته به مدل. نرمال‌سازی دسته یک تکنیک است که با آن فعال‌سازی‌های یک شبکه عصبی نرمال شوند، که می‌تواند بهبود سرعت و استقرار آموزش را فراهم کند.

کلاس "Activation" از ماژول "keras.layers" توابع فعال‌سازی غیرخطی را به مدل اضافه می‌کنند که به آن امکان یادگیری الگوها و ارتباطات پیچیده در داده را می‌دهد.

کلاس "MaxPooling2D" از ماژول "keras.layers" برای افزودن لایه‌های ماکس‌پولینگ به مدل.

کلاس "AvgPool2D" از ماژول "keras.layers" برای افزودن لایه‌های اوریج‌پولینگ به مدل.

کلاس "Flatten" از ماژول "keras.layers" برای تغییر شکل تنسور ورودی به یک تنسور یک‌بعدی. تغییر شکل تنسور، تنسور چند بعدی را به یک تنسور یک‌بعدی تبدیل می‌کند که قبل از ارسال داده به یک لایه کاملاً متصل ضروری است.

کلاس "Dense" از ماژول "keras.layers" برای افزودن لایه‌های کاملاً متصل به مدل. لایه‌های کاملاً متصل برای انجام پیش‌بینی‌ها بر اساس ویژگی‌های استخراج شده توسط لایه‌های کانولوشن استفاده می‌شوند.

تابع "L2" از ماژول "keras.regularizers" برای افزودن منظم‌سازیL2. تنظیم L2 تکنیکی است که با افزودن یک عبارت جریمه به تابع ضرر که مدل را تشویق می‌کند وزن‌های کوچکی داشته باشد، برای جلوگیری از بیش‌برازش استفاده می‌شود.

کلاس "Adam" از ماژول "keras.optimizers" برای استفاده از بهینه‌ساز Adam . Adam یک الگوریتم بهینه سازی است که مزایای هر دو الگوریتم AdaGrad و RMSProp را ترکیب می‌کند و معمولاً برای آموزش شبکه‌های عصبی عمیق استفاده می شود.

کتابخانه‌های pd و np برای انجام عملیات مربوط به داده‌ها و تعامل با آنها. این کتابخانه‌ها توابعی برای کار با داده‌ها در قالب جدولی فراهم می‌کنند، مانند بارگذاری و تغییر داده‌ها.

ماژول kutils از کتابخانه keras.utils توابع کمکی مختلفی مربوط به Keras را ارائه می‌دهد، مانند-one hot encodingو ارزیابی مدل.

ماژول scores از کتابخانه sklearn.metrics استفاده شده است تا معیارهای ارزیابی مختلفی مانند،دقت (precision)، بازخوانی (recall) و امتیاز F1 را محاسبه کند.

ماژول tqdm یک Progress Bar برای نمایش پیشرفت یک حلقه یا یک تکرار فراهم می کند.

plotly.graph_objects یک ماژول است که واسط کاربری سطح بالا برای ایجاد و به‌روزرسانی نمودارها فراهم می‌کند. این امکان را به ما می‌دهد تا انواع مختلفی از نمودارها را ایجاد کنیم، از جمله نمودارهای پراکندگی، نمودارهای میله‌ای و نمودارهای خطی و غیره.

ماژول plotly.graph_objects برای رسم نمودارها و تصویرسازی داده‌ها به کار رود.

plotly.offline نیز یک ماژول است که امکانات آفلاین برای Plotly فراهم می‌کند. این به ما اجازه می‌دهد تا نمودارهای Plotly را بسازیم و آنها را بدون نیاز به اتصال اینترنت ایجاد و ذخیره کنیم.

**پردازش و فراخوانی اولیه داده‌ها:**

ابتدا تمام دایرکتوری‌های موجود در دایرکتوری "./dataset/" را با استفاده از تابع os.listdir() لیست می‌شود. سپس یک لیست به نام eligible_dirs ایجاد می‌شود که به هر نام دایرکتوری، زیرمسیر "./dataset/" را اضافه می‌کند.

پس از آن، یک دیکشنری خالی به نام images مقداردهی اولیه می‌شود. سپس برای هر دایرکتوری در eligible_dirs حلقه می‌زند و نتیجه next(os.walk(directory))[2] را به کلید images[directory] اختصاص می‌دهد. تابع os.walk() یک ژنراتور بازمی‌گرداند که برای هر دایرکتوری یک تاپل تولید می‌کند. عنصر دوم این تاپل لیست نام‌های فایل در آن دایرکتوری است. در این حالت، next(os.walk(directory))[2] لیست نام‌های فایل در دایرکتوری فعلی را بازمی‌گرداند.

سپس، یک لیست به نام subsets تعریف می‌شود که شامل پنج زیرلیست است. هر زیرلیست یک زیرمجموعه از اندیس‌ها را نمایان می‌سازد.

سپس، کد یک pandas DataFrame به نام subset_dataframe ایجاد می‌کند. توضیح لیست تاپل‌های حاوی اندیس و برچسب برای هر عنصر در لیست subsets است. متغیر اندیس نمایانگر اندیس عنصر در زیرلیست خود است و متغیر برچسب نمایانگر اندیس زیرلیست در لیست subsets است. لیست نهایی از تاپل‌ها سپس به تابع pd.DataFrame() منتقل شده و DataFrame ایجاد می‌شود. سپس DataFrame بر اساس ستون "index"با استفاده از متد sort_values() مرتب می‌شود.

در نهایت، کد DataFrame subset_dataframe را خروجی می‌دهد.

در ادامه کد یک تابع به نام load تعریف می‌شود که یک پارامتر مسیر را می‌پذیرد. این تابع مسئول بارگیری یک تصویر از مسیر مشخص شده و بازگرداندن آن به صورت یک آرایه numpy است.

تابع ابتدا از تابع preprocess.image.load_img برای بارگیری تصویر از مسیر مشخص شده استفاده می‌کندو باید با حالت رنگ RGB بارگیری شود. علاوه بر این، تعیین شده است که تصویر باید به ابعاد مقصد 160 x 160پیکسل تغییر اندازه داده شود که در متن پروژه ذکر شده است.

تصویر بارگیری شده سپس با استفاده از تابع preprocess.image.img_to_array به یک آرایه numpy تبدیل می‌شود. این تابع تصویر را به یک آرایه 3 بعدی تبدیل می‌کند که هر پیکسل توسط یک بردار سه مقدار (قرمز، سبز و آبی) نمایان می‌شود. آرایه حاصل بر 255 تقسیم می‌شود تا مقادیر پیکسل بین 0 و 1 نرمالیزه شوند.

در نهایت، تابع آرایه numpy نتیجه نمایانگر تصویر بارگیری شده را بازمی‌گرداند.

**تعریف مدل شبکه عصبی(Neural Network Model Definition): [3]**

در این قسمت از کد دو کلاس تعریف می‌شود ConvBlock و CustomModel.

کلاس ConvBlockیک بلوک کانولوشنی در یک شبکه عصبی را نمایش می‌دهد. این کلاس دارای ویژگی‌هایی مانند filters ( تعداد فیلترها در لایه کانولوشنی)، kernel_size (اندازه هسته کانولوشنی)، strides (شاب)، padding، pool_size (اندازه پنجره پولینگ)، pool_strides (شاب عملیات پولینگ) و regularizer است.

همچنین، کلاس ConvBlock دارای یک متد به نام add_to_model است که یک model را به عنوان ورودی می‌گیرد و بلوک کانولوشنی را به مدل اضافه می‌کند. این متد یک لایه کانولوشنی با ویژگی‌های مشخص شده اضافه می‌کند، سپس به آن batch normalization و فعال‌سازی ReLU را اضافه می‌کند. و در نهایت، یک لایه ماکس‌پولینگ نیز اضافه می‌شود.

کلاس CustomModel یک مدل شبکه عصبی سفارشی را نمایش می‌دهد. این کلاس دارای ویژگی‌هایی مانند model (مدل Keras)، l2_regularization، initial_learning_rate، num_classes (تعداد کلاس‌های خروجی) و input_shape (شکل داده ورودی) است.

کلاس CustomModelدارای یک متد به نام build است که مدل را ایجاد می‌کند. این متد با اضافه کردن یک لایه کانولوشنی با 16 فیلتر، اندازه هسته (3، 3) و ویژگی‌های مشخص شده شروع می‌شود. سپس یک لیست از نمونه‌های ConvBlockبا ویژگی‌های مختلف ایجاد شده و با استفاده از متد add_to_model به مدل اضافه می‌شوند.

پس از اضافه کردن بلوک‌های کانولوشنی، مدل flatten می‌شود و یک لایه Dense با 256 واحد اضافه می‌شود، در ادامه از فعال‌ساز ReLU استفاده می‌شود در نهایت، یک لایه Dense با تعداد مشخص شده کلاس‌های خروجی و فعال‌ساز softmax اضافه می‌شود.

به طور خلاصه این یک مدل شبکه عصبی سفارشی با بلوک‌های کانولوشنی و لایه‌های Dense تعریف می‌کند. کلاس ConvBlock امکان سفارشی‌سازی آسان بلوک‌های کانولوشنی را فراهم می‌کند و کلاس CustomModelراهی مناسب برای ساختن مدل با ویژگی‌های مشخص شده ارائه می‌دهد.

در قسمت پایانی این بخش از کد compile در کلاس CustomModel که مسئول تنظیم تمامی پارامترهای مرتبط با آموزش مدل است و این تنظیمات شامل انتخاب بهینه‌ساز، تابع هزینه و معیارها ارزیابی مدل می‌شود تنظیم می‌شود:

optimizer=Adam(learning_rate=self.initial_learning_rate): این بخش مشخص می‌کند که از الگوریتم بهینه‌ساز Adam برای آموزش استفاده شود و نرخ یادگیری آن با مقدار self.initial_learning_rate تعیین شود.

- loss='categorical_crossentropy': این بخش تعیین می‌کند که برای محاسبه خطا از تابع هزینه categorical_crossentropy برای مسائل دسته‌بندی چند دسته‌ای استفاده شود.
- metrics=['accuracy']: در این قسمت، معیار(ها) ارزیابی برای آموزش مدل مشخص می‌شود و در اینجا (accuracy) انتخاب شده است.

## Data Augmentation Setup:

این قسمت از کد برای تولید داده‌های افزوده برای وظایف دسته‌بندی تصویر استفاده می‌شود. از کلاس ImageDataGenerator از ماژول preprocess.image استفاده می‌شود.

کلاس ImageDataGenerator با چندین پارامتر مقداردهی اولیه می‌شود که نوع و دامنه تبدیلاتی که بر تصاویر ورودی اعمال می‌شود را تعریف می‌کنند. این تبدیلات شامل چرخش، تغییرات افقی و عمودی، تبدیلات شیر، بزرگنمایی و وارونه کردن هستند.

مقداردهی پارامتر ها بدین شکل است[2]:

```
rotation_range=20          # Random rotation between 0 and 20 degrees
width_shift_range=0.2      # Horizontal shift up to 20% of the width
height_shift_range=0.2     # Vertical shift up to 20% of the height
shear_range=0.2            # Shear transformations up to 20%
zoom_range=0.2             # Zooming up to 20%
horizontal_flip=True       # Horizontal flip
vertical_flip=True         # Vertical flip (disabled)
```

پس از مقداردهی اولیهImageDataGenerator ، کد دو حالت تصادفی از یک DataFrame به نام yalb_dataframe انتخاب می‌کند. از روش sample برای انتخاب تصادفی دو مقدار از ستون 'person' DataFrame استفاده می‌شود و از روش tolist برای تبدیل مقادیر انتخاب شده به یک لیست استفاده می‌شود. سپس، با استفاده از متد isin، DataFrameبر اساس حالات انتخاب شده فیلتر می‌شود. این متد یک ماسک بولین ایجاد می‌کند که نشان می‌دهد آیا هر مقدار در ستون 'person' در لیست حالات انتخاب شده قرار دارد یا خیر. این ماسک برای فیلتر کردن DataFrame استفاده می‌شود و یک DataFrame جدید به نام data ایجاد می‌شود که فقط حاوی سطرهای متناظر با حالات انتخاب شده است.

بعد از آن، کد برچسب‌های کلاس را به DataFrame فیلتر شده اختصاص می‌دهد. برای هر حالت انتخاب شده، با استفاده از تابع enumerate برای دریافت همزمان اندیس و مقدار هر حالت، برای هر حالت از تابع loc برای

انتخاب سطرهای DataFrame data استفاده می‌کند که ستون 'person' برابر با مقدار فعلی حالت است. سپس اندیس متناظر با برچسب کلاس برای این سطرها اختصاص داده می‌شود.

در نهایت، کد DataFrame data را بازمی‌گرداند که حالا داده‌های افزوده با برچسب‌های کلاس اختصاص یافته را شامل می‌شود.

**معرفی داده‌های آموزش و آزمون(Training and Test data):**

در این قسمت با توجه به صورت پروژه کد مجموعه داده را به دو قسمت تقسیم می‌کند: مجموعه آزمون و مجموعه آموزش.

اولاً، تمام سطرهای مجموعه داده که ستون 'label' آنها مقدار 2 دارد را انتخاب می‌کند و به متغیر 'test' اختصاص می‌دهد.

سپس، تمام سطرهای مجموعه داده که ستون 'label' آنها مقدار 2 ندارد را انتخاب می‌کند و به متغیر 'train' اختصاص می‌دهد.

متغیرهای 'test' و 'train' سپس چاپ می‌شوند، که مجموعه داده‌های متناظر نمایش داده می‌شوند.

سپس، دو آرایه به نام‌های 'train_class' و 'test_class' ایجاد می‌شوند با استخراج مقادیر ستون 'class' از مجموعه‌های 'train' و'test'.

به مشابه، دو آرایه به نام‌های 'train_data' و 'test_data' ایجاد می‌شوند با استخراج مقادیر ستون 'filepath' از مجموعه‌های 'train' و'test' .

در نهایت، مقادیر 'test_data' و 'train_data' چاپ می‌شوند، که مسیرهای فایل مجموعه آزمون و آموزش را به ترتیب نمایش می‌دهند.

سپس plot Loss /epoch جهت ارزیابی عملکرد مدل نمایش داده می‌شود.

در ادامه مقدار Accuracy,Loss محاسبه شده و نمایش داده می‌شود.

### پیش‌بینی مجموعه آزمون ( Test Set predict and Scores):

این بخش از کد برای پیش‌بینی مجموعه آزمون و محاسبه معیارهای عملکرد مانند دقت(precision) ، فراخوانی (recall)و F1 Score استفاده می‌شود.

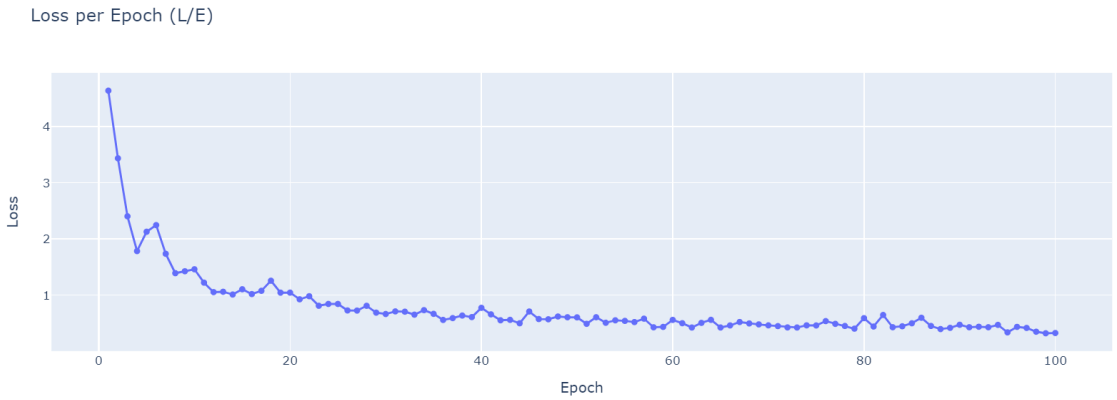
اولاً، از یک مدل سفارشی برای پیش‌بینی برچسب‌ها برای داده‌های آزمون استفاده می‌شود. تابع custom_model.model.predict() با داده‌های آزمون (test_1.X_data) به عنوان ورودی فراخوانی می‌شود. این تابع احتمالات پیش‌بینی شده برای هر کلاس در داده‌های آزمون را برمی‌گرداند. سپس از تابع np.argmax() برای پیدا کردن اندیس کلاس با بیشترین احتمال برای هر نقطه داده استفاده می‌شود، که به تبع آن برچسب‌های پیش‌بینی شده حاصل می‌شود.

سپس، با استفاده از توابع scores.precision_score()، scores.recall_score() و scores.f1_score() از ماژول scores، precision ، recall و F1 Score محاسبه می‌شود. این توابع با گرفتن برچسب‌های کلاس واقعی (test_class) و برچسب‌های پیش‌بینی شده به عنوان ورودی، معیارهای عملکرد مربوطه را محاسبه می‌کنند. پارامتر average='weighted' برای محاسبه میانگین وزن‌دار معیارها استفاده می‌شود.

در نهایت معیارهای عملکرد در یک دیکشنری به نام performance_metrics ذخیره می‌شوند.

## خروجی‌های حاصل (Result):

خروجی ۱:




شکل۲) خروجی۱ نمودار L/E

<< Train >>

Best Accuracy: 0.9574

Worst Accuracy: 0.5319

Best Loss: 0.3189

Worst Loss: 4.6392

<< Test >>

Accuracy= 1.0000

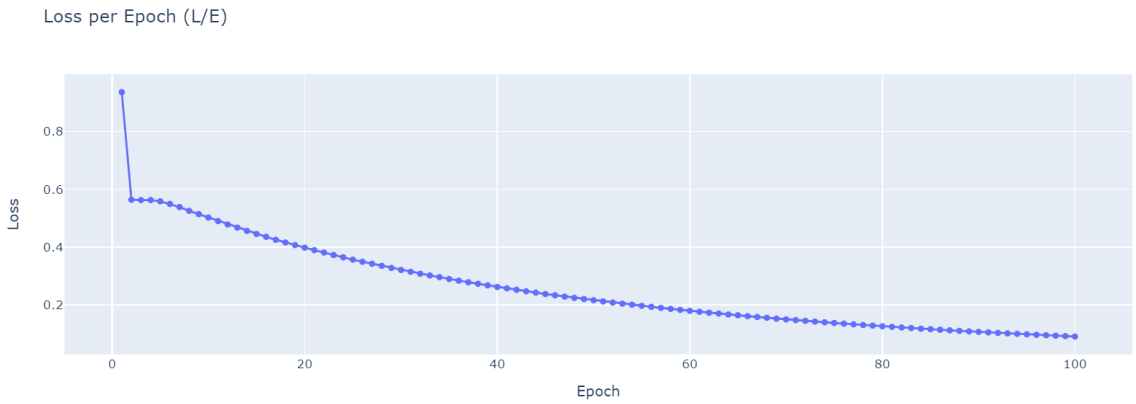
Loss= 0.19214089214801788

Precision=1

Recall=1

F1 Score=1

خروجی ۲:




شکل۳) خروجی۲ نمودار L/E

<< Train >>

Best Accuracy: 1.0000
Worst Accuracy: 0.8140
Best Loss: 0.0910
Worst Loss: 0.9355

<< Test >>

Accuracy= 1.0000

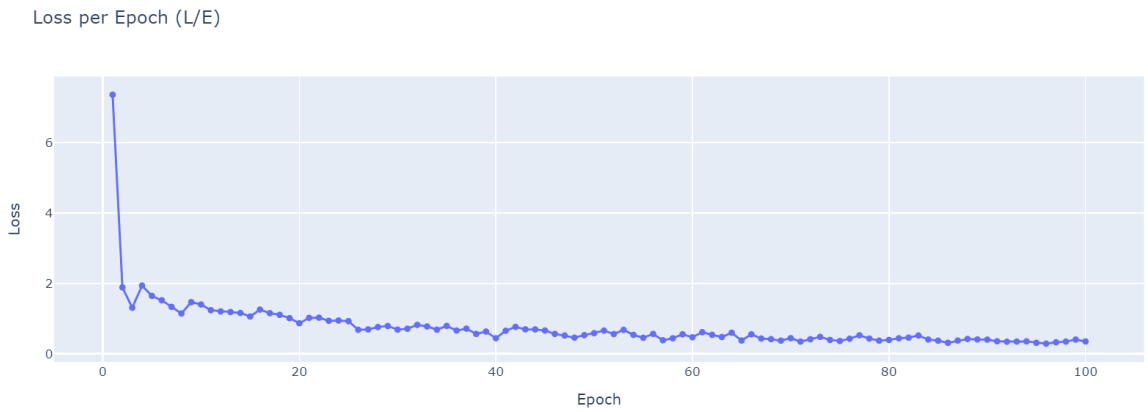
Loss= 0.09013137221336365

Precision=1

Recall=1

F1 Score=1

خروجی ۳:




شکل۴) خروجی۳ نمودار L/E

<< Train >>

Best Accuracy: 0.9468
Worst Accuracy: 0.4681
Best Loss: 0.2855
Worst Loss: 7.3613

<< Test >>

Accuracy= 0.75

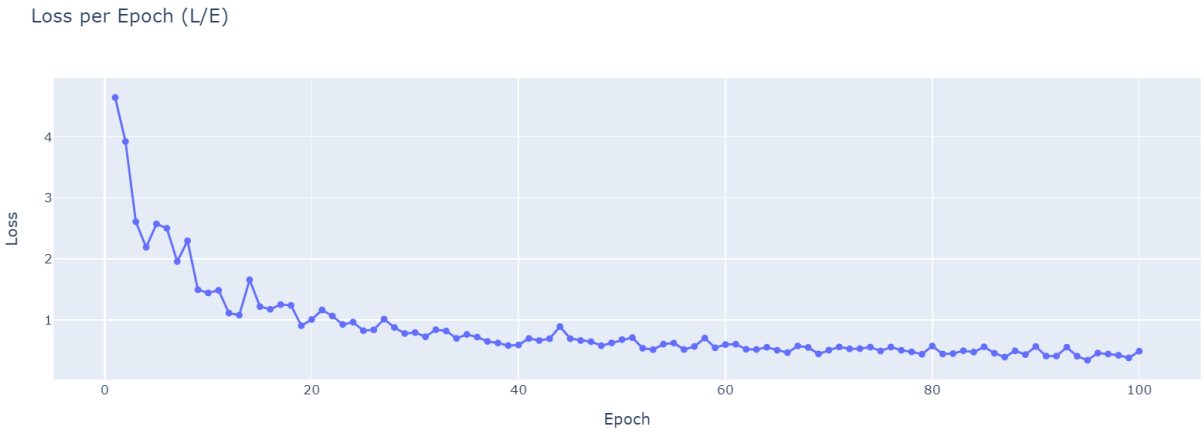
Loss= 0.7482870221138

Precision= 0.8333333333333334

Recall= 0.75

F1 Score= 0.7333333333333334

خروجی ۴:




شکل۵) خروجی۴ نمودار L/E

< Train >>

Best Accuracy: 0.9362
Worst Accuracy: 0.5213
Best Loss: 0.3398
Worst Loss: 4.6441

<< Test >>

Accuracy= 0.8333333134651184

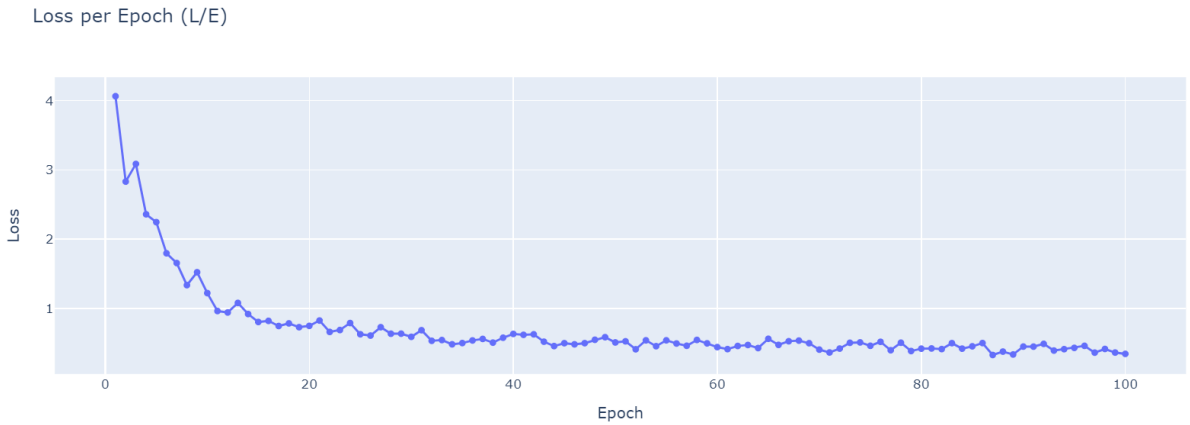
Loss= 0.4711252450942993

Precision= 0.875

Recall= 0.8333333333333334

F1 Score= 0.8285714285714286

خروجی ۵:




شکل۶) خروجی۵ نمودار L/E

< Train >>

Best Accuracy: 0.9681
Worst Accuracy: 0.4681
Best Loss: 0.3269
Worst Loss: 4.0647

<< Test >>

Accuracy= 0.875

Loss= 0.524488627910614

Precision= 0.875

Recall= 0.875

F1 Score= 0.873015873015873

**جمع‌بندی :**

در این گزارش و پروژه، ارزیابی جامعی از یک مدل شبکه عصبی کانولوشنی (CNN) برای دسته‌بندی تصاویر در دسته‌های مختلف مجموعه داده صورت گرفت. هدف اصلی ارزیابی عملکرد مدل از نظر دقت، خطا، (Precision)، (Recall) و (F1 Score) بود. مشاهدات و نتایج کلیدی زیر را می توان از نتایج تجربی از آزمایش‌های مختلف انجام شده استخراج کرد:

- **تغییرات عملکرد در دسته‌های مختلف:**
  - مدل در طول فرآیند آموزش توانست دقت و خطاهای متفاوتی را در دسته‌های مختلف نشان دهد.
  - ویژگی‌ها و پیچیدگی‌های مرتبط با هر دسته تاثیرگذار بر نتایج به دست آمده در طی آموزش بودند.
- **ارزیابی روی مجموعه آزمون:**
  - مدل دقت بالایی در مجموعه آزمون ارائه کرد، که نشان‌دهنده کارآیی مدل در تعمیم به داده‌های ناشناخته است.
  - معیارهای Precision ، Recall و F1 Score روی مجموعه آزمون، قابلیت مدل در حفظ عملکرد متوازن در معیارهای مختلف را تاکید کرد.
- **پایداری مدل و چالش‌ها:**
  - هرچند مدل در مجموعه آزمون عملکرد بسیار خوبی داشت، اما عملکرد متفاوتی در دسته‌های خاص در طول آموزش، چالش‌های مرتبط با داده‌های متنوع و پیچیده را نشان داد.
  - استفاده از تکنیک‌های افزایش داده به پایداری مدل کمک کرده است، اما نیاز به تنظیم دقیق‌تر ممکن است برای بهبود تطابق مدل با دسته‌های گوناگون باشد.
- **حساسیت به هایپرپارامترها:**
  - هایپرپارامترهایی نظیر اندازه دسته، تعداد epoch ، نرخ یادگیری ابتدایی و (L2 Regularization) نقش‌های اساسی در شکل‌دهی به رفتار مدل داشتند.
  - بهینه‌سازی مداوم و نظارت بر این پارامترها ممکن است منجر به بهبود عملکرد کلی مدل شود.

## منابع و مراجع

[1] https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

[2] https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/#h-random-rotations

[3] https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python/

[4] https://medium.com/machine-learning-researcher/convlutional-neural-network-cnn-2fc4faa7bb63

[5] https://plotly.com/python/getting-started/