به نام خدا

دانشگاه شهید بهشتی

دانشکده علوم ریاضی

گزارش تمرین دو شبکه عصبی

زهرا دهقانی تفتی (۹۶۲۲۲۰۳۷)

آبان ۹۹

فهرست مطالب

علائم ترافیکی

راه‌حل و ایده‌های کلی

در این مسئله هدف ما این است تا با دریافت یک تصویر از علائم راهنمایی رانندگی بتوانیم تشخیص دهیم تا برچسب این تصویر کدام است و نام آن را با استفاده از مدلی که می‌سازیم تشخیص دهیم و در واقع با یک مسئله‌ی کلاس بندی چند کلاسه با ٤٣ کلاس سر و کار داریم.

در این مسئله داده‌های ما متشکل از ٤ فایل است که عبارتند از : ٣ فایل مربوط به عکس ها که فایل دادهای آموزشی، اعتبار سنجی و تست می‌باشند و یک فایل مربوط به برچسب علائم راهنمایی رانندگی است. در فایل عکس‌ها ٤٣ نوع تابلوی راهنمایی رانندگی وجود دارد و فایل عکس‌ها به صورت یک دیکشنری به ما داده شده‌است.

هدف ما این است تا مدلی را بسازیم تا با گرفتن یک تصویر بتواند برچسب آن را تشخیص دهد. مدلی که ما برای اینکار انتخاب می‌کنیم convolutional neural network است. دلیل ما از انتخاب این مدل این است که cnn برای کلاس بندی تصاویر بسیار مناسب است زیرا بر خلاف mlp، تحمل تغییرپذیری در داده‌های ورودی را دارد مثل بزرگ یا کوچک شدن عکس، بالا یا پایین رفتن عکس و ... پس بهترین مدل برای اینکار است زیرا تصویرهایی که به ما داده می‌شود ممکن است کیفیت مناسب نداشته باشد یا هر کدام از علائم در جای فیکسی از تصویر نباشند.

در ابتدا کاری که باید بکنیم این است که کتابخانه های مورد نیاز برای اینکار را فراخوانی کنیم، سپس به داده‌های خود در گوگل درایو دسترسی پیدا کنیم و داده‌ها را لود کنیم.

در ابتدای کار با دو چالش مواجه می‌شویم و باید برای آموزش موفقیت آمیز کلاس بندی دقیق علائم راهنمایی رانندگی آزمایشی را طراحی کنیم که بتواند این دو چالش را حل کند و چالش‌ها عبارتند از: ١- پیش پردازش عکس‌های ورودی برای بهبود بخشیدن به فرآیند مقایسه چون مثلا ممکن است عکس‌ها در شرایط نوری یکسانی نباشند و همین امر باعث شود قسمت‌هایی از عکس روشن‌تر و غیرقابل تشخیص شوند و برای مقایسه مناسب نباشند. ٢- درست کردن عدم توازن کلاس ها به این معنی که امکان دارد در داده‌های آموزشی ما برای مثال تعداد خیلی زیادی از تصاویر متعلق به کلاس ٣و ٤ باشند و بقیه‌ی کلاس‌ها مقدار کمی داده داشته باشند همین امر باعث می‌شود وقتی مدل را روی داده‌های آموزشی آموزش دادیم، مدلمان این گونه یاد بگیرد که بیشتر عکس‌ها مربوط به کلاس ٣ و ٤ هستند و موقع تست کردن احتمال اینکه داده‌های زیادی را به اشتباه در کلاس ٣ یا ٤ کلاس بندی کند زیاد باشد زیرا عدم توازن داده‌ها در کلاس‌ها باعث شده تا جواب‌ها به سمت کلاس خاصی bias داشته باشد.

برای دیدن چالش اول ابتدا در کد تعداد تصویر از هر علامت راهنمایی رانندگی چاپ شده. برای حل این چالش باید روی تصاویر ورودی پردازش‌هایی انجام شود. ابتدا باید پیکسل‌های ورودی را به عدد ٢٥٥ تقسیم کنیم تا نرمال شوند و بهتر قابل مقایسه گردند. دلیل این تقسیم بر عدد ٢٥٥ این است که هر پیکسل در تصویر می‌تواند عدد ٠ تا ٢٥٥ را بگیرد و این نشان دهنده‌ی شدت رنگ در تصویر است. هرچه پیکسل روشن‌تر باشد مقداری که میگیرد بیشتر است و ٠ مربوط به رنگ مشکی است.

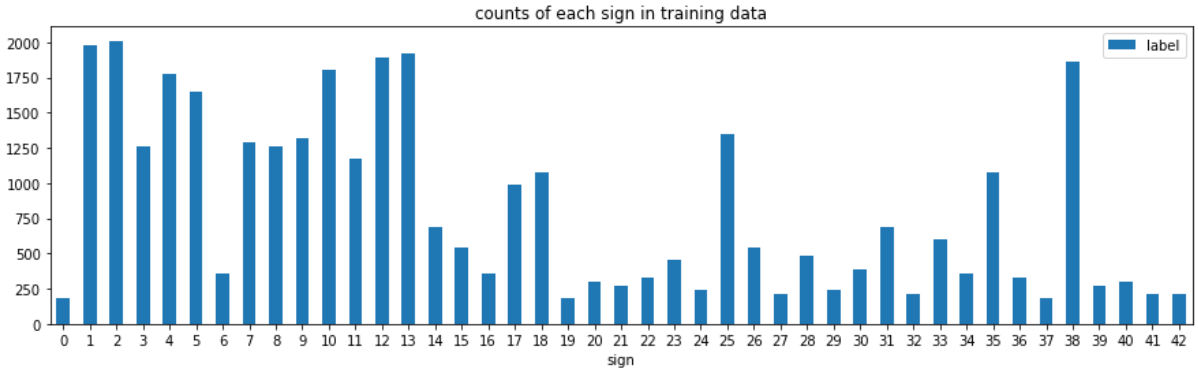
بعد از آن باید y ها برای داده‌های آموزشی، اعتبار سنجی و تست به نوع دسته بندی تبدیل شوند زیرا ما در اینجا ۴۳ کلاس داریم و برای هر y باید به categorical تبدیل شوند یعنی خود y به یک لیست یا آرایه‌ی ۴۳ تایی تبدیل شود. برای مثال اگر خروجی برچسب شماره ۱۲ بود باید عنصر ۱۲ ام این لیست ۱ شود. و برای y های پیش بینی هر کدام از این ۴۳ عنصر، مقدار دارند ولی بزرگترین عنصر به معنای بیشترین احتمال حضور در آن کلاس می‌باشد که بعدا می‌بینیم این بیشترین احتمال را با استفاده از تابع argmax متوجه می‌شویم.

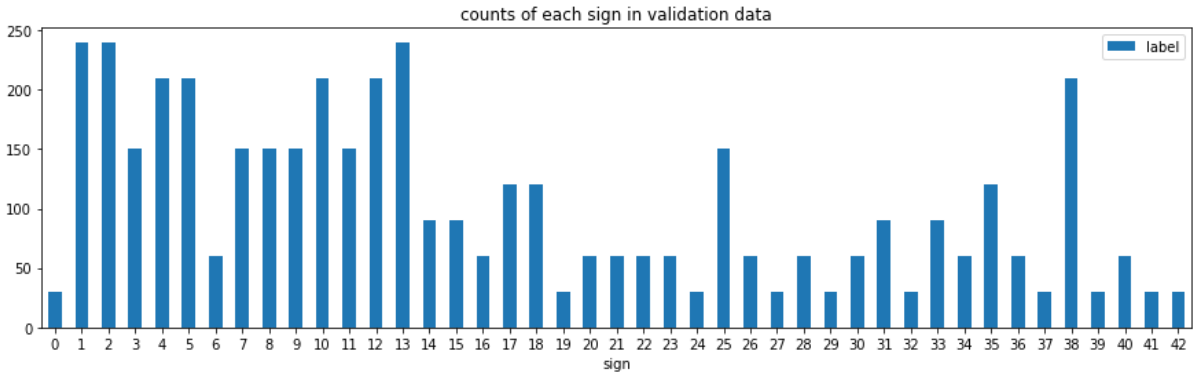
بعد از این کار برای برطرف کردن مشکل نور عکس ها از تابعی به اسم histogram equalization استفاده می‌کنیم. این تابع روشی برای پردازش تصاویر به منظور تنظیم کنتراست تصویر با اصلاح توزیع شدت هیستوگرام است. هدف این تکنیک ، دادن یک روند خطی به تابع احتمال تجمعی مرتبط با تصویر است به گونه‌ای که مشکل نور عکس را برطرف کند و برای مثال مکان‌هایی را که شدت نور در آن زیا است و قابل تشخیص نیستند را بهبود دهد.

این تابع را در تابعی به نام Imagedatagenerator قرار می‌دهیم. هدف از به کار بردن Imagedatagenerator تولید تصاویر بیشتر از تصاویر موجود برای مدل به منظور افزایش دقت و تعمیم پذیری مدل است و یک روش عالی برای افزایش اندازه مجموعه داده خواهد بود.

کاری که این تابع انجام می‌دهد این است که از تصاویر موجود استفاده می‌کند تا تصاویر جدیدی تولید کند تا ابعاد مختلفی از تصاویر موجود را داشته باشیم. این کار به مدل ما کمک می‌کند تا داده‌های بیشتری را ببیند و از آن‌ها اطلاعات استخراج کند و باعث افزایش دقت در مدل می‌شود. در واقع تصاویر جدید تعدادی از همان تصاویر اصلی هستند که زوم شده‌اند، چرخیده‌اند، بالا پایین یا چپ و راست شده‌اند و ...

تصاویر زیر چالش دوم را نشان می‌دهند.این تصاویر به ترتیب نشان دهنده‌ی توزیع داده‌های آموزشی واعتبارسنجی در ۴۳ دسته هستند.





همانطور که در شکل مشخص است توزیع داده‌ها در کلاس‌ها نامتوازن است و برای حل این مشکل و جلوگیری از bias به سمت کلاس خاص در پیش بینی، باید به هرکلاس یک وزن داده شود که در نهایت این وزن‌ها در fit کردن مدل به‌کار گرفته می‌شوند.

بعد از این کار وارد طراحی معماری برای مدل برای مسئله می‌شویم. همانطور که گفتیم برای این مسئله از cnn استفاده می‌شود. ساختار cnn به این صورت است که در ابتدا ویژگی ها را با استفاده از توالی لایه های convolutional و pooling استخراج می‌کند و سپس با استفاده از شبکه Fully connected که در انتهای آن است کار کلاس بندی را انجام می‌دهد.

تابع فعالیتی که برای لایه‌های conv مورد استفاده قرار می‌گیرد معمولا Relu است. تابع فعالیت برای لایه‌های آخر در شبکه که fully connected هستند معمولا Softmax یا sigmoid است زیرا این دو تابع برای کلاس بندی مناسب هستند و یک ایندکس یا عدد گسسته‌ای را به ما می‌دهند .

چون در این مسئله تعداد کلاس‌های ما ۴۳ تا است و بیشتر از ۲ است از Softmax استفاده می‌کنیم که در لیستی از احتمال پیش بینی برای هر کلاس را داشته باشیم و بیشترین عدد نشان دهنده‌ی بیشترین احتمال حضور در آن کلاس است.

برای دادن خروجی لایه های conv به fully connected از لایه‌ی flatten استفاده می‌کنیم که داده‌ها را به صورت بردار یک بعدی قابل قبول برای fully connected در می‌آورد.

از لایه‌های batch normalization برای نرمال سازی دسته‌ای استفاده می‌شود و روشی برای سریع‌تر کردن و پایدارتر کردن شبکه‌های عصبی مصنوعی از طریق نرمال سازی لایه ورودی با استفاده مجدد از مرکز و مقیاس بندی مجدد است.

در اینجا ما ابتدا مدل را روی داده‌های عکس رنگی آموزش می‌دهیم و بعد از بدست آوردن بهترین مدل، آن مدل را روی عکس‌های سیاه سفید امتحان می‌کنیم تا مقایسه کنیم روی کدام بهتر عمل می‌کند و آیا رنگی بودن در دقت تاثیرگذار است یا نه.

تبدیل به تصاویر در مقیاس خاکستری ضمن حفظ ویژگی های جالب مانند اشکال و نمادها ، از پیچیدگی داده ها می‌کاهد. اگر ما انسانها بتوانیم با این اطلاعات کم شده علائم راهنمایی و رانندگی را برچسب گذاری کنیم ، انتظار داریم شبکه عصبی نیز

بتواند این کار را انجام دهد. در حقیقت ، کاهش ابعاد از ۳ رنگ به ۱، علاوه بر کارآیی و آموزش آسان‌تر، می‌تواند عملکرد را به طور بالقوه افزایش دهد، زیرا رنگ گاهی منبع مشکلات است و باعث "حواس پرتی" می شود. تصاویر با استفاده از تابع rgb2gray به مقیاس خاکستری تبدیل شدند.

در انتهای پروژه برای اینکه ببنیم مدل روی چه قسمت‌هایی از عکس تمرکز دارد از تابعی به اسم Grad Cam استفاده می‌کنیم. این تابع نحوه‌ی بدست آوردن فعالسازی نقشه گرمایی کلاس برای یک مدل طبقه بندی تصاویر را نشان می‌دهد و بیانگر این است که در چه قسمت‌هایی از عکس تمرکز داشته است.

در این پروژه از یک call back به اسم checkpointer استفاده می‌کنیم که در هر اجرای مدل، وزن بهترین epoch را ذخیره می‌کند تا اگر در هنگام آموزش مدل، وقفه ایجاد شد مشکلی پیش نیاید و تا آن epoch وزن بهترین مدل ذخیره شده باشد.

برای

ارزیابی نتایج

آزمایش ۱:

برای این مسئله آزمایش از ساختار زیر استفاده می‌کنیم . در مدل اول برای توازن کلاس‌ها، به کلاس‌ها وزن نمی‌دهیم تا ببینیم این عدم توازن در کلاس‌های مختلف چه تاثیری دارد و آیا باعث bias در نتایج و کم شدن دقت می‌شود یا نه.

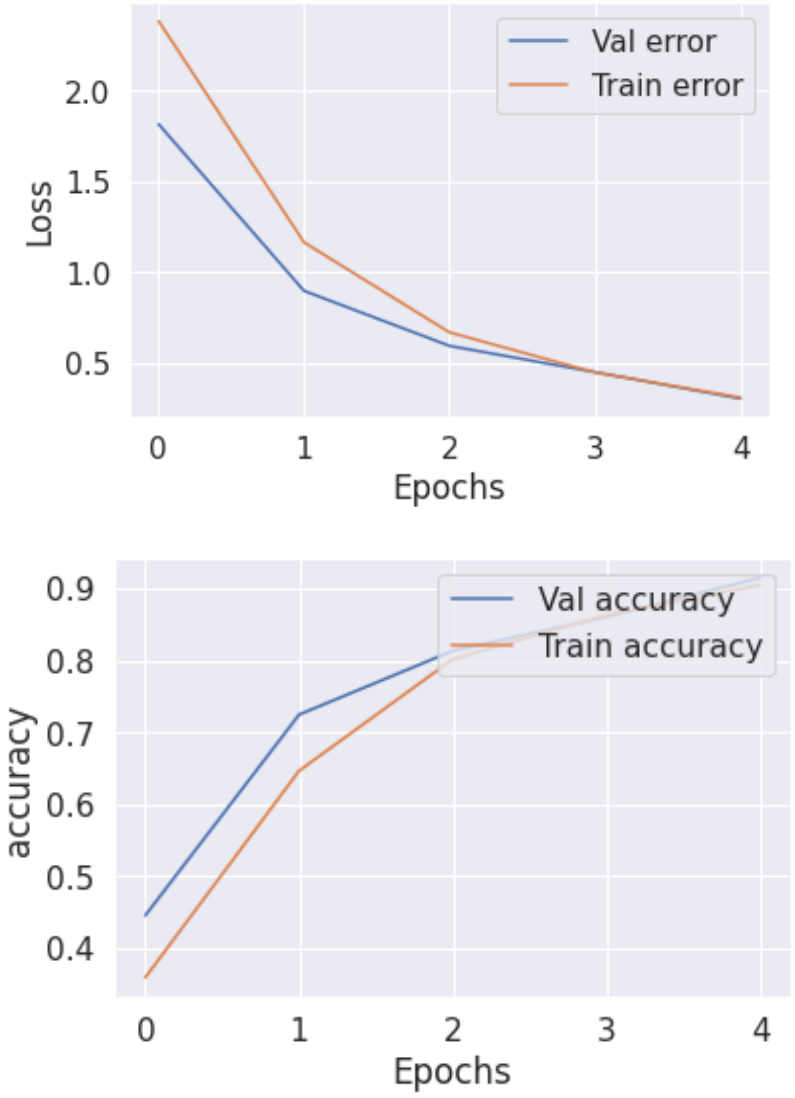
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	[(None, 32, 32, 3)]	0
Conv1_5 (Conv2D)	(None, 28, 28, 32)	2432
Bnorm1 (BatchNormalization)	(None, 28, 28, 32)	128
Conv2_5 (Conv2D)	(None, 28, 28, 32)	25632
Bnorm2 (BatchNormalization)	(None, 28, 28, 32)	128
MaxPool1 (MaxPooling2D)	(None, 14, 14, 32)	0
Conv3_3 (Conv2D)	(None, 14, 14, 64)	18496
Bnorm3 (BatchNormalization)	(None, 14, 14, 64)	256
Conv4_3 (Conv2D)	(None, 14, 14, 64)	36928
Bnorm4 (BatchNormalization)	(None, 14, 14, 64)	256
AvgPool1 (AveragePooling2D)	(None, 7, 7, 64)	0
Conv5_1 (Conv2D)	(None, 7, 7, 128)	8320
Bnorm5 (BatchNormalization)	(None, 7, 7, 128)	512
Flatten (Flatten)	(None, 6272)	0
dense_3 (Dense)	(None, 32)	200736
dropout_1 (Dropout)	(None, 32)	0
batch_normalization_3 (Batch	(None, 32)	128
dense_4 (Dense)	(None, 20)	660
batch_normalization_4 (Batch	(None, 20)	80
dense_5 (Dense)	(None, 15)	315
batch_normalization_5 (Batch	(None, 15)	60
preds (Dense)	(None, 43)	688

Total params: 295,755
Trainable params: 294,981
Non-trainable params: 774

در لایه‌های ابتدایی شبکه تعدا فیلتر کمتری قرار می‌دهیم زیرا می‌خواهیم ویژگی‌های ساده‌تری را مورد بررسی قرار دهیم اما هرچقدر عمیق‌تر شویم مایلیم ویژگی‌های پیچیده‌تری مثل تشخیص کل تابلو را بررسی کنیم پس باید تعداد فیلترهای conv که استفاده می‌کنیم بیشتر شود. برای لایه pooling هم سایز ۲ را درنظر می‌گیریم تا مطمئن شویم اطلاعاتی از دست نمی‌رود. این مدل را با ۵ ایپاک و بهینه‌گر SGD و استفاده از momentum برای تسریع در رسیدن به نقطه‌ی بهینه اجرا می‌کنیم و به نتیجه‌ی زیر می‌رسیم. طبق خطا روی داده‌های آموزشی بنظر می‌رسد مدل نتوانسته به خوبی داده‌ها را یاد بگیرد و احتمالا به دلیل عدم توازن کلاس‌ها می‌باشد.

```
loss: 0.3116 - accuracy: 0.9050 - val_loss: 0.3058 - val_accuracy: 0.9152
```

نمودارهای زیر به ترتیب نمودار خطا و دقت بر روی داده‌های آموزشی و اعتبار سنجی هستند.



آزمایش ۲:

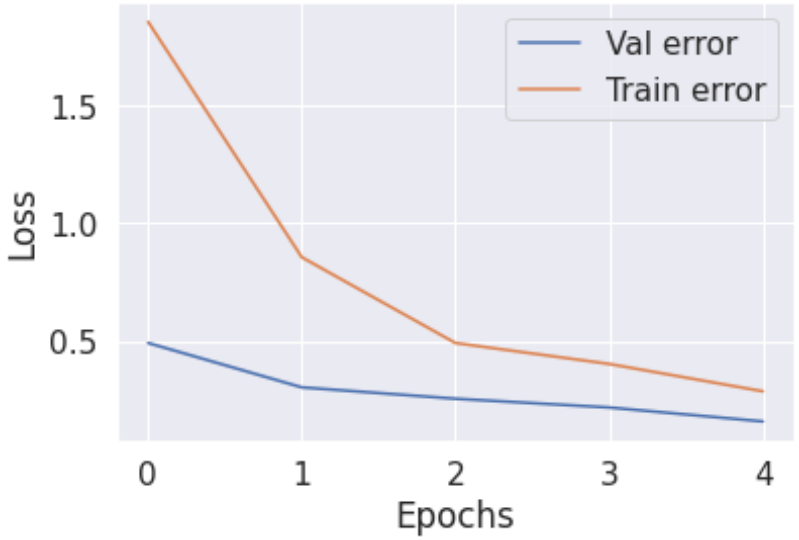
این مدل دقیقا مشابه مدل ۱ است با این تفاوت که در این جا کلاس‌ها را با دادن وزن به آن‌ها متوازن کرده‌ایم و انتظار داریم نتیجه بهتری بگیریم.

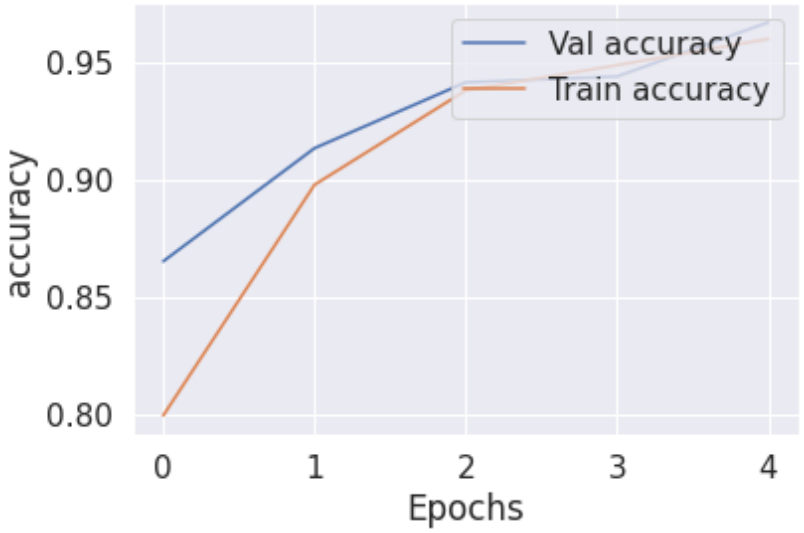
این مدل را اجرا می‌کنیم و به نتیجه زیر می‌رسیم.

```
loss: 0.2883 - accuracy: 0.9600 - val_loss: 0.1604 - val_accuracy: 0.9671
```

در این مدل دقت هم روی داده‌های آموزشی هم داده‌های اعتبار سنجی افزایش پیدا کرده‌است و خطا نیز کاهش پیدا کرده واین بخاطر متوازن شدن وزن‌ها است و به همین باعث شده پیش بینی‌های دقیق تری داشته باشیم.

نمودارهای زیر به ترتیب نمودار خطا و دقت بر روی داده‌های آموزشی و اعتبار سنجی هستند.





آزمایش ۳:

در این آزمایش تعداد لایه‌ها و ساختار ما مشابه مدل ۲ است زیرا مدل دو تا حالا بهترین مدل ما بوده است. روی مدل ۲ یک تغییر می‌دهیم و آن این است که فیلتر لایه‌ی conv اول بجای اینکه ۵*۵ باشد ۳* ۳ در نظر می‌گیریم.معماری این مدل به شکل زیر است.

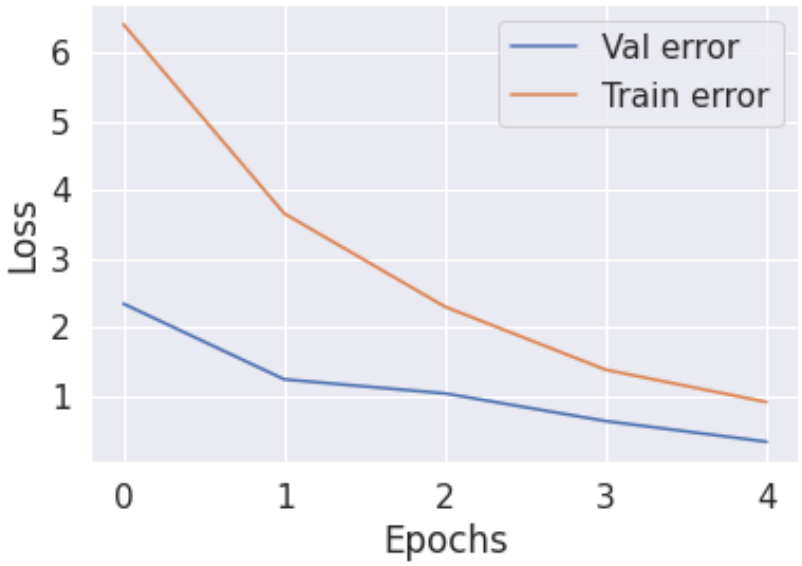
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	[(None, 32, 32, 3)]	0
Conv1_5 (Conv2D)	(None, 30, 30, 32)	896
Bnorm1 (BatchNormalization)	(None, 30, 30, 32)	128
Conv2_5 (Conv2D)	(None, 30, 30, 32)	25632
Bnorm2 (BatchNormalization)	(None, 30, 30, 32)	128
MaxPool1 (MaxPooling2D)	(None, 15, 15, 32)	0
Conv3_3 (Conv2D)	(None, 15, 15, 64)	18496
Bnorm3 (BatchNormalization)	(None, 15, 15, 64)	256
Conv4_3 (Conv2D)	(None, 15, 15, 64)	36928
Bnorm4 (BatchNormalization)	(None, 15, 15, 64)	256
AvgPool1 (AveragePooling2D)	(None, 7, 7, 64)	0
Conv5_1 (Conv2D)	(None, 7, 7, 128)	8320
Bnorm5 (BatchNormalization)	(None, 7, 7, 128)	512
Flatten (Flatten)	(None, 6272)	0
dense_6 (Dense)	(None, 32)	200736
dropout_2 (Dropout)	(None, 32)	0
batch_normalization_6 (Batch	(None, 32)	128
dense_7 (Dense)	(None, 20)	660
batch_normalization_7 (Batch	(None, 20)	80
dense_8 (Dense)	(None, 15)	315
batch_normalization_8 (Batch	(None, 15)	60
preds (Dense)	(None, 43)	688

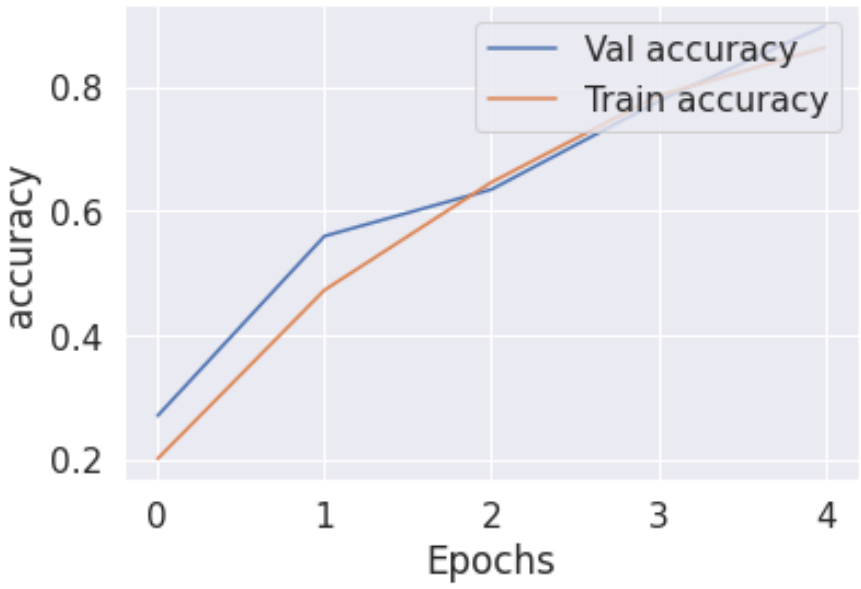
Total params: 294,219
Trainable params: 293,445
Non-trainable params: 774

با این مدل به نتیجه زیر می‌رسیم.

```
loss: 0.9035 - accuracy: 0.8628 - val_loss: 0.3259 - val_accuracy: 0.8977
```

می‌بینیم که نتیجه نسبت به مدل ۲ بهبود پیدا نکرده و دلیل آن این است که شاید لزومی به داشتن فیلترهای کوچک تر نباشد و این کار باعث می‌شود که آموزش مدل بیشتر طول بکشد (تقریبا هر epoch ما ۱ ثانیه بیشتر در این مدل نسبت به مدل دو طول می‌کشد.) و دقت و خطا نسبت به مدل ۲ بهبود پیدا نکند. در واقع بنظر می‌رسد وجود فیلتر ۵ در ۵ برای لایه‌ی اول برای بررسی ویژگی‌های ساده یک عکس در ای مسئله کافی باشد و نیازی به وجود فیلتر ۳در ۳ و بررسی دقیق‌تر نباشد. به عبارتی ویژگی‌های تصویر ما طوری است که فیلتر ۵ در ۵ باعث از دست رفتن ویژگی از آن نمی‌شود و همه ویژگی‌های آن را بررسی می‌کند.





ارزیابی روی داده‌های تست : طبق آزمایش‌های انجام شده مدل ۲ بهترین مدل ما است. حال می‌خواهیم این مدل را روی داده‌های تست بکار ببریم تا ببینم دقت و خطای مدل روی داده‌های تست چقدر است. با لود کردن مدل روی داده‌های تست به نتیجه زیر می‌رسیم.

```
loss: 1.2275 - accuracy: 0.8011
```

این نشان می‌دهد که دقت روی داده‌های تست ۸۰ درصد است و بنظر می‌رسد خوب باشد.

این دو خط کد زیر نتیجه پیش بینی شده روی داده‌های تست را نشان می‌دهد و چون تابع فعالیت ما Argmax است عنصری از لیست خروجی ما که بیشترین احتمال را داشته باشد، index آن به عنوان برچسب برای عکس استفاده می‌شود.

```
predict_test = model.predict(test_list)
predict_test_id = predict_test.argmax(axis=1)
```

بعد از این کار برای درک بهتر، تعدادی از تصاویر تست را همراه با برچسب پیش بینی شده چاپ می‌کنیم

برای اینکه بدانیم چه داده‌هایی بیشتر با هم اشتباه گرفته می‌شوند confusion matrix را برای مدل روی داده‌های تست رسم می‌کنیم.دراین ماتریس داده‌ها در ستون نشان دهنده‌ی برچسب‌های واقعی و داده‌ها در سطر نشان دهنده‌ی برچسب‌های پیش بینی شده برای تصاویر هستند.

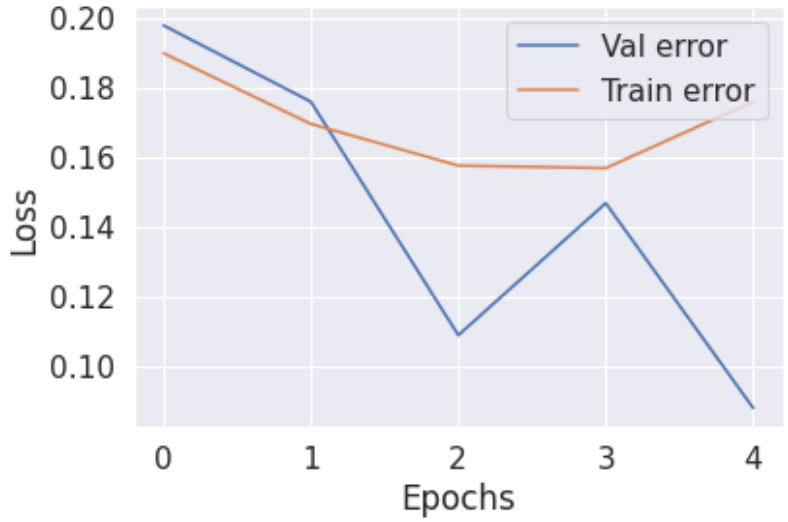
بعد از رسم این ماتریس میبینیم که اکثریت تابلوها درست تشخیص داده شده‌اند؛ زیرا قطر این ماتریس پر رنگ است و داده‌های زیادی را شامل می‌شود و این به معنی این است که برچسب واقعی و پیش بینی شده مانند هم هستند. ولی یک سری از تابلوها هستند که با هم زیاد اشتباه گرفته می‌شوند برای مثال ٦٠ تابلوی خطر انحراف به چپ در داده‌های تست است که فقط ۳۰ تای آن درست تشخیص داده شده‌است. ۱۵ تای آن با تابلوی محدودیت عبور اشتباه گرفته شده و ۱۲ تای بقیه با تابلوی عبور کودکان اشتباه گرفته شده.

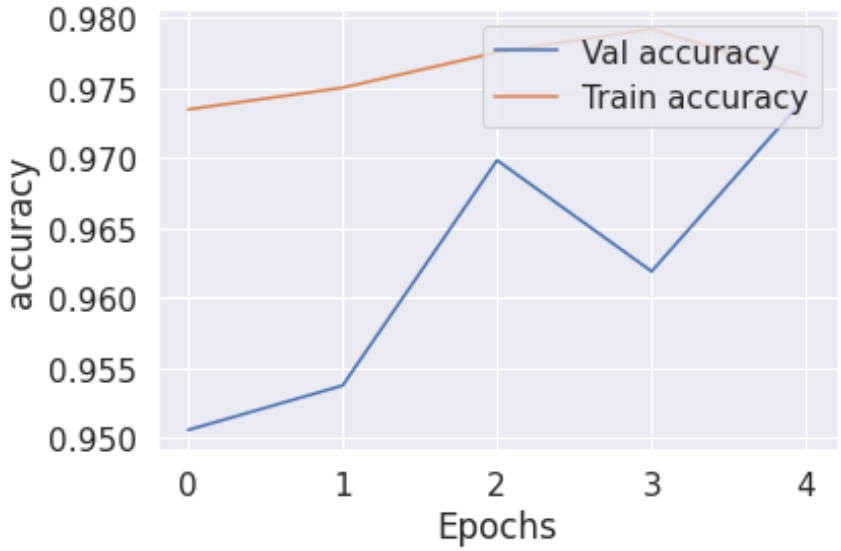
یا مثلا برای تابلوی حق تقدم حدودا نصف این تابلوها با تابلوی بدون وسیله نقیله اشتباه گرفته شده. برای اینکه علت این اشتباه را تشخیص دهیم و بدانیم تمرکز شبکه ما برای تشخیص عکس به کدام قسمت عکس است و شبکه گرمایی آن را تشخیص دهیم از الگوریتم grad cam استفاده می‌کنیم. که نمونه‌های استفاده از آن و تحلیل عکس‌ها در نوت بوک هست.

بعد از این کار عکس‌هایمان را به سیاه وسفید تبدیل می‌کنیم تا پیچیدگی آن کم شود سپس با استفاده از این عکس‌ها ساختار مدل دوم زا آموزش می‌دهیم تا ببینیم چه تاثیری دارد و با عکس رنگی مقایسه کنیم. وقتی عکس‌ها سیاه و سفید شوند زمانی که برای آموزش مدل صرف می‌شود تا ٤ برابر کاهش پیدا می‌کند و این نشان دهنده‌ی این است که پیچیدگی آموزش ما کمتر شده است. با انجام اینکار به نتیجه زیر می‌رسیم که نتیجه خیلی بهتری نسبت به داده‌های رنگی روی داده‌های اعتبار سنجی میگیریم هم از لحاظ خطا و هم از لحاظ دقت.

```
loss: 0.1758 - accuracy: 0.9758 - val_loss: 0.0880 - val_accuracy: 0.9744
```

شکل‌های زیر نشان دهنده‌ی نمودارهای مربوط به این مدل هستند.





با اجرا کردن این مدل روی داده‌های تست میبینیم که دقت نسبت به حالت رنگی کاهش پیدا کرده ولی تفاوت خیلی زیادی حس نمی‌شود و در بار قبلی که من این مدل را اجرا کردم دقت روی داده‌های سیاه سفید نسبت به داده‌های رنگی بیشتر بود ولی در بار بعدی دقت روی داده سیاه سفید کمی کمتر شد. که این نشان دهنده‌ی این است که رنگی بودن عکس‌ها تاثیری در دقت ندارد و این تابلوها را می‌توان در حالت سیاه و سفید نیز تشخیص داد و بهتر هم هست چون از پیچیدگی داده‌ها کم می‌شود و در حافظه و زمان صرفه جویی می‌شود در حالی که استفاده از داده‌های رنگی ٤ برابر بیشتر نسبت به این حالت زمان می‌برد و در مورد حافظه برای عکس‌های رنگی باید هر ۳ کانال عکس نگهداری شود ولی برای عکس‌های سیاه سفید باید فقط یک کانال نگهداری شود که خب هزینه کمتری دارد و در گاهی موارد رنگی بودن عکس سبب حواس پرتی می‌شود که در این موارد استفاده از عکس سیاه سفید به صرفه‌تر است

جمع‌بندی و نتیجه‌گیری

طبق آزمایش‌های انجام شده نتیجه می‌گیریم که اگر در کلاس‌ها عدم توارن وجود داشته باشد باید رفع گردد تا باعث گیج شدن مدل در هنگام یادگیری نشود و نتیجه‌ی بهتری بگیریم.

اگر برای تنیم شرایطی نوری عکس‌ها از تابع histogram equalization استفاده شود سرعت آموزش مدل چند برابربیشتر می‌شود نسبت به زمانی که از تابع adaptive histogram equalization برای این‌کار استفاده شود.

در شرایطی که تشخیص و کلاس بندی عکس‌ها وابسته به رنگ آن‌ها نمی‌باشد از لحظ حافظه و زمان به صرفه‌تر است که از عکس‌های سیاه سفید استفاده شود تا پیچیدگی مدل کمتر شود و سرعت یادگیری و احتمالا دقت بالاتر رود و طبق تحقیقات من در سایت‌های مختلف، نتایج تجربی نشان داده‌اند که کلاس بندی با تصاویر در مقیاس خاکستری منجر به کلاس بندی با دقت بالاتری نسبت به تصاویر رنگی در انواع کلاس بندی‌ها می‌شود و می‌توان با انتخاب ویژگی‌های لبه-ای در تصاویر خاکستری هزینه محاسباتی را نیز کاهش داد.

در ادامه‌ی کد خروجی لایه‌های مختلف شبکه را روی عکس دلخواه چاپ می‌کنیم. همچنین اینکار را برای فیلترها هم می‌کنیم و فیلترهای لایه‌های conv را چاپ می‌کنیم تا بفهمیم چه ویژگی هایی را بررسی می‌کنند.فیلترها در واقع ماتریس‌های وزن هستند که برحسب اینکه یک پیکسل روشن تر باشد یعنی مقدار فیلتر در این نقطه بیشتر است و اگر پیکسل تیره تر باشد یعنی مقدار وزن در این نقطه کمتر است.(عدد صفر نمایانگر رنگ سیاه و عدد ۲۵۵ نمایانگر سفید است)

در مورد خروجی لایه‌های مختلف conv هر نقطه که روشن تر باشد یعنی تطابق بیشتری با فیلتر داشته که عکس روشن تر شده و مقدار پیکسل بیشتری گرفته است. در لایه‌های ابتدایی ویژگی‌های ساده‌ای مثل وجود خط بررسی می‌شوند ولی هرچه عمیق‌تر شویم ویژگی‌های ساده‌تر با هم ترکیب می‌شوند و ویژگی‌های پیچیده‌تری بررسی می‌شوند. Feature map هایی که سیاه هستند نشان دهندی این هستند که با فیلتر مورد نظرشان اصلا تطابقی نداشتند.

Transfer learning : برای بهبود نتیجه مدل‌ها می‌توان از مدل‌های آماده که دقت خیلی خوب و خطای کمی دارند استفاده کرد مثل alexnetو...

به این روش transfer learning می‌گویند و به این صورت است که مدل cnn آماده‌ی ما گرفته می‌شود و لایه‌های conv آن فیریز هستند و تغییری نمی‌دهیم و تنها در لایه‌های Fully connected آن که برای کلاس بندی و یا رگرسیون استفاده می‌شود، تغییر می‌دهیم.

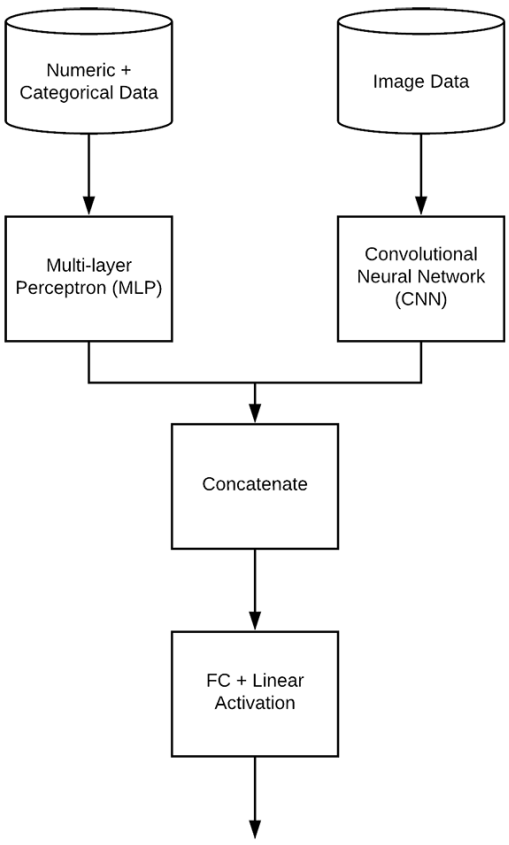
تخمین قیمت خانه

راه حل و ایده های کلی

در این مسئله هدف ما این است تا با دریافت تصویر و اطلاعات مربوط به یک خانه قیمت این خانه را پیش بینی کنیم. این مسئله از نوع مسائل رگرسیون می‌باشد و ما باید بر حسب یک تعداد داده‌ای که از قبل داریم یک خروجی که در اینجا قیمت است را برای داده‌های تست پیش بینی کنیم.

این مسئله چند ورودی دارد و ورودی‌های آن از انواع مختلفی می‌باشند. ورودی‌های این مسئله به صورت عکس ، عددی و متن می‌باشند. هر عکس و اطلاعات مربوط به آن ورودی مدل ما هستند و خروجی یک عدد است که قیمت آن خانه می‌باشد.

ورودی عکس باید به یک مدل convolutional neural network داده شود و ورودی‌ها متنی و عددی باید به multi layer perceptron داده شوند سپس خروجی cnn و mlp باهم ترکیب شوند و به یک fully connected داده شود تا قیمت پیش بینی شود. در واقع باید مدلی به شکل زیر را طراحی کنیم.



برای این کار ابتدا باید داده‌ها را دریافت کنیم و سپس پردازش و عملیات مورد نیاز را را روی آن‌ها انجام دهیم. ابتدا داده‌های متنی و عددی را از فایل CSV می‌خوانیم. تعداد داده‌های ما ۱۵۴۷۴ است و ۸ ویژگی متنی و عددی داریم. ابتدا باید داده‌ها را تمیز کنیم . بررسی کنیم که آیا در ستون‌ها مقادیر خالی وجود دارد یا نه. اگر مقادیر خالی وجود داشته باشد یک کار ممکن

است که سطر مربوط به آن داده حذف شود و راه دیگر این است که آن مقدار خالی را با استفاده از میانگین آن ویژگی در داده‌ها پر کنیم. همانطور که در کد مشخص هست در این کد مقداز خالی یا null نداریم.

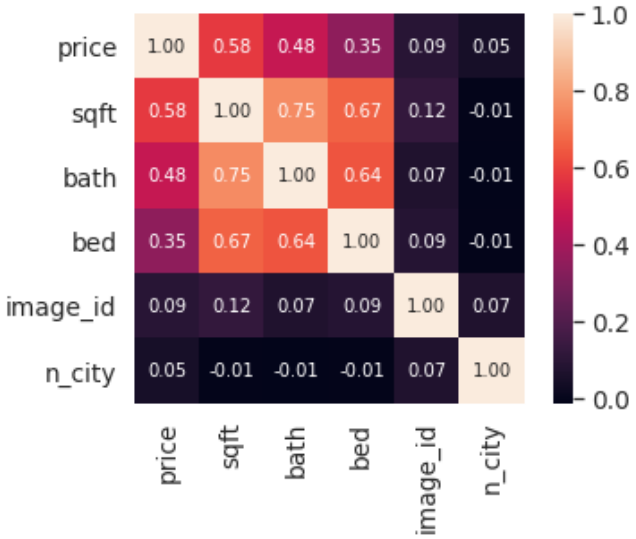
با استفاده از دستور nunique می‌توان تعداد مقادیر یکتایی که هر ویژگی می‌تواند داشته باشد را فهمید. برای مثال ویژگی bed می‌تواند ۱۲ مقدار مختلف داشته باشد.

بعد از آن برای درک بهتر داده‌ها می‌توان با استفاده از دستور describe، داده‌ها را توصیف کرد و کمترین مقدار، بیشترین مقدار، چارک اول، چارک دوم و... را مشخص نمود.

بعد از آن برای درک اینکه قیمت‌ها در چه حدودی هستند هیستوگرام قیمت را برای داده‌ها رسم می‌کنیم. تمودار زیر نشان دهنده‌ی این است که قیمت بیشتر خانه‌ها در حدود ۲۵۰۰۰۰ تا ۱۰۰۰۰۰۰ هستند. این نشان دهندی این است که احتمالا قیمت بیشتر خانه‌ها در این بازه پیش بینی می‌شود چون پیش بینی ما براساس داده‌های قبلی هست و همین احتمال وجود خطا در پیش بینی را به دنبال دارد و یک کار ممکن اسن است که عکس‌ها و اطلاعاتی که مربوط به خانه‌هایی با قیمت بیشتر از ۱۰۰۰۰۰۰ در داده‌هایمان وجود دارد را حذف کنیم تا دقت بالاتر رود

بعد از این تاثیر ویژگی‌های مختلف در قیمت را با استفاده از ماتریس وابستگی به نمایش می‌گذاریم. هدف از اینکار این است که ویژگی‌هایی که تاثیر زیادی در قیمت گذاری ندارند و نامربوط هستند، حذف شوند تا بتوانیم نتیجه‌ی بهتری بگیریم.

طبق ماتریس وابستگی صفحه بعد می‌بینیم که مساحت خانه تاثیر زیادی در قیمت دارد، پس ازآن تعداد حمام‌ها و پس از آن تعداد اتاق‌ها بیشترین تاثیر را در قیمت خانه دارند. هرچقدر خانه‌های این ماتریس روشن‌تر باشند نشان دهنده‌ی وابستگی بیشتر بین سطر و ستون آن خانه از ماتریس است. در این ماتریس شماره عکس بیشتر از شماره خیابان در قیمت تاثیر دارد ولی ما می‌دانیم که شماره عکس هیچ اولویتی در قیمت ایجاد نمی‌کند و صرفا نشان دهنده‌ی داده‌های مربوط به یک عکس است. در این ماتریس فقط ویژگی‌های عددی را بررسی کرده است و ویژگی‌هایی مانند خیابان و شهر را که از نوع object هستند را بررسی نمی‌کند.



طبق این جدول و بررسی‌هایی که شد شماره عکس را باید از داده‌ها حذف شود زیرا هیچ تاثیری در قیمت ندارد و هر داده شماره عکس متفاوتی دارد.

همچنین داده‌هایی که از نوع string هستند و در این ماتریس نبودند هم باید بررسی شوند. ویژگی‌های خیابان و شهر نیز در تعیین قیمت خانه مهم هستند زیرا قیمت هر خانه منطقه به منطقه فرق دارد. برای اینکه این دده‌ها به صورت قابل فهم برای شبکه عصبی دربیاید باید آن‌ها را به نوع دسته‌بندی تبدیل کنیم و با استفاده از label encoding به هر خیابان برچسبی نسبت داده شود و از dummy variable برای شهر استفاده شود به اینصورت که به تعداد شهرهای متمایز ستون به داده‌ها افزوده می‌شود و هر شهر که انتخاب شود ستون مربوط به آن مقدار یک می‌گیرد که در اینصورت باعث می‌شود شهرها وزن دار نباشند ب اینصورت که شماره شهرها با عث تغییری در قیمت ایجاد نکنند و همه‌ی شهرها از نظر شبکه یکسان بنظر برسند و وزن خاصی نداشته باشند . (برای مثال اگر یک شهر برچسب ۲ بگیرد و شهر دیگر برچسب ۳ این امکام وجود دارد که شبکه به اشتباه بیفتد و قیمت خانه‌هایی را که در شهر ۳ هستند را بیشتر پیش بینی کند زیر ۳<۲ در صورتی که منظ.ر ما این نبوده به خاطر همین از dummy variables استفاده می‌شود.)

سپس برای مرتب شدن بهتر داده‌ها ون حمام‌ها قسمت اعشاری دارند و نشان دهنده‌ی halfbath می‌باشد این قسمت اعشاری را جدا می‌کنیم پس یک سطر دیگر به جدول اضافه می‌شود.

تا اینجای کار همه‌ی ویژگی‌های عددی و متنی به صورت قابل فهم برای شبکه درآمدند. ویژگی شماره شهر نیز باید حذف شود زیرا هیچ تاثیری در قیمت خانه ندارد و وجود ویژگی شهر کافی است. پس تمام ویژگی‌های غیرمربوط نیز حذف شدند.

بعد اینکار باید داده‌های مربوط به ویژگی‌ها و خروجی جدا شوند. در اینجا خروجی ستون مربوط به قیمت است.

بعد از خواندن داده‌های متنی و عددی نوبت به خواندن و load کردن داده‌های عکس‌ها می‌باشد.

بعد خواند داده‌های مربوط ب عکس باید عکس‌های ناقص تاجای ممکن جذف گردند. بریا مثال یکسری از عکس‌های موجود در داده‌های ما هنوز لود نشده‌اند و ٤ کاناله هستند و یا یک تعدادی عکس خانه نیستند و مربوط به عکس location است.

حذف این عکس‌ها و اطلاعات مربوط به آن‌ها در جدول اطلاعات متنی به دقت مدل ما کمک می‌کند و باعث می‌شود پیش بینی‌های دقیق‌تری داشته باشیم.

همچنین در داده‌های متنی ستون مربوط به street برای برخی از داده‌ها Address not provided است. این نیز جز مواردی است که اگر حذف شود به دقت مدل ما و یادگیری بهتر آن کمک می‌کند.

چون سایز عکس‌ها یکسان نیست و مد نمی‌تواند سایزهای مختلف را دریافت کند پس باید عکس‌ها را تغییر سایز دهیم و یک سایز کنیم.

بعد از اینکار باید عکس‌ها و ویژگی‌ها و به همراه خروجی را به دو دسته تست آموزشی تقسیم کنیم و داده‌ها را نرمال کنیم.

بعد از اینکار نوبت به آموزش مدل می‌رسد. ورودی مدل ما یک عکس به همراه اطلاعات مربوط به آن می‌باشد و خروجی آن قیمت خانه است.

چون مسئله‌ی ما رگرسیون است و با مقادیر پیوسته سرو کار داریم؛ متریک‌هایی که برای اینکار استفاده می‌شود با حالت کلاس بندی فرق دارد و برای تابع خطا نیز نمی‌توان از cross entroy یا binary entropy استفاده کرد. در این مدل‌ها ما برای خطا از تابع mean squared error استفاده می‌کنیم.

همچنین چون مدل ما رگرسیون است برای تابع فعالیت لایه‌ی آخر باید از linear استفاده کرد و لایه‌ی آخر فقط باید شامل یک نورون باشد.

برای اینکه اهمیت تصاویر در پیش‌بینی خانه را با بقیه‌ی ویژگی‌ها مقایسه کنیم باید یکبار mlp طراحی کنیم و فقط داده‌های متنی و عددی را به آن بدهیم و یکبار cnn طراحی کنیم و فقط داده‌های عکس را به ان بدهیم. و در نهایت تابع خطا برای داده‌های اعتبارسنجی را برای این دو مدل با هم مقایسه کنیم تا بفهمیم کدام یک از این دو تاثیر بیشتری در پیش بینی قیمت خانه دارد و اهمیت آن بیشتر است.

کاری که ابتدا انجام می‌دهیم این است که یک مدل با چند ورودی و وردی‌هایی از انواع مختلف طراحی کنیم و سپس داده‌های متنی و عکسی را به آن بدهیم. در مدل‌هایی که طراحی کردم خطا روی داده‌های اعتبار سنجی و تست در طی ٦ اپاک کاهش پیدا نمی‌کند و بنظر می‌رسد که مدل ما vanish شده است و گرادیان به خاظر وجود لایه‌های زیاد به سمت صفر میل می‌کند بنابراین وزن‌ها تغییر پیدا نمی‌کند و در نتیجه خطا نیز کم نمی‌شود.