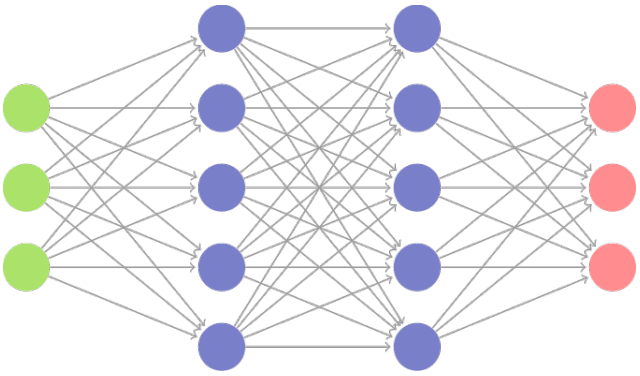
به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



Neural Networks

Project 2

نام و نام خانوادگی	علیرضا محمدی	محمدعلی شاکردرگاه
شماره دانشجویی	۸۱۰۱۹۹۳۶۵	۸۱۰۱۹۶۴۸۷
تاریخ ارسال گزارش	۱۴۰۰/۰۴/۰۴	

فهرست مطالب

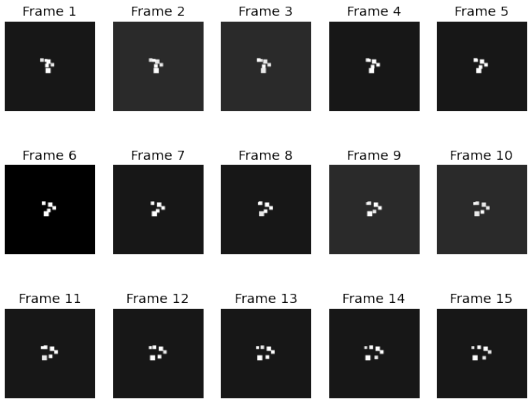
۱ سوال دوم

در این بخش می‌خواهیم با استفاده از مدل کانولوشنی به همراه مدل بازگشتی، مدلی را برای پیش‌بینی فریم‌های یک ویدیو توسعه دهیم. هدف آن است تا بتوانیم با استفاده از تعدادی از فریم‌های یک ویدیو، فریم‌های بعدی را پیش‌بینی کنیم.

۱.۱ دیتاست

دیتاست استفاده شده در این سوال به وسیله‌ی تابع `generate_movies` ایجاد می‌شود. این تابع تعداد دلخواهی ویدیو را که هر یک تعداد مشخصی فریم دارند، ایجاد می‌کند. در این بخش ما به تعداد ۳۰۰ دیتا که هر کدام ۲۱ فریم دارند را ایجاد می‌کنیم.

ویدیوهای ایجاد شده توسط این تابع، شامل یک زمینه‌ی مشکی است که تعدادی مربع با اندازه‌های تصادفی (بین ۳ تا ۸ مربع به تصادف) در آن ترسیم شده‌اند و هر کدام با یک سرعت و راستای تصادفی حرکت می‌کنند.



شکل ۱: فریم‌های یک دیتا از این دیتاست

۲.۱ طراحی مدل

در طراحی مدل برای پیش‌بینی فریم بعدی در یک ویدیو، دو راهکار وجود دارد. راهکار اول این است که با استفاده از شبکه کانولوشنی، ویژگی‌های هر فریم را استخراج کرده و سپس خروجی این شبکه را به یک بردار تبدیل کرده و نهایتا از این بردار به عنوان ورودی شبکه بازگشتی استفاده کنیم. سپس می‌توانیم خروجی شبکه بازگشتی را در هر قدم زمانی، با استفاده از یک شبکه کانولوشنی معکوس به تصویر تبدیل کنیم.

راهکار دوم آن است که از لایه‌های کانولوشنی بازگشتی استفاده کنیم.در این لایه‌ها به جای استفاده از لایه fully connected در لایه بازگشتی، از لایه‌های کانولوشنی استفاده می‌شود. در این بخش هر دو راهکار مورد بررسی قرار خواهند گرفت.

۱.۲.۱ ساختار مدل

برای شبکه بر مبنای راهکار اول، از ساختار زیر استفاده می‌کنیم:

۱. یک لایه کانولوشنی دو بعدی با ۳۲ فیلتر، استراید ۲ و تابع فعال‌سازی ReLU.

۲. یک لایه کانولوشنی دو بعدی با ۱ فیلتر، استراید ۲ و تابع فعال‌سازی ReLU.

۳. لایه Flatten برای تبدیل خروجی لایه کانولوشنی به بردار.

۴. لایه بازگشتی LSTM با ۹۶۱ واحد مخفی.

۵. لایه Reshape برای تبدیل بردار خروجی لایه بازگشتی به یک تصویر دو بعدی.

۶. لایه معکوس کانولوشنی با ۳۲ فیلتر و استراید ۲.

۷. لایه معکوس کانولوشنی با ۳۲ فیلتر و استراید ۲.

۸. لایه کانولوشن معکوس سه بعدی با استراید ۲.

تمامی لایه‌های فوق به جز لایه‌های ۴ و ۸، با استفاده از لایه‌ی TimeDistributed تعریف شده‌اند تا این فیلترها بر روی تمامی فریم‌های دیتا اعمال می‌شود. تعداد فیلترها در لایه دوم به دلیل زیاد بودن تعداد پارامترهای لایه بازگشتی، به ازای ورودی‌های بزرگ‌تر ، برابر با ۱ انتخاب شده‌است. زیرا به ازای افزایش تعداد فیلترها، حجم پارامترهای بین لایه سوم و چهارم به شدت افزایش پیدا می‌کند که در روند آموزش اثر منفی می‌گذارد. این لایه در واقع یک pooling را بر روی ۳۲ تصویر خروجی لایه اول انجام می‌دهد و در عین کاهش تعداد تصاویر به یک تصویر، کاهش بعد نیز انجام می‌شود.

ورودی شبکه که یک تصویر ۲۵۰ در ۲۵۰ پیکسل است، پس از گذر از لایه‌های اول و دوم، به یک تصویر ۶۲ در ۶۲ پیکسل تبدیل می‌شود. سپس این تصویر با تبدیل به یک بردار به لایه بازگشتی داده می‌شود. نهایتا خروجی این لایه با گذر از ۳ لایه کانولوشنی دو بعدی به یک ماتریس ۱۲۵ در ۱۲۵ در ۳۲ تبدیل می‌شود که با اعمال یک کانولوشن سه‌بعدی بر روی تمامی فریم‌های دیتا، یک تصویر ۲۵۰ در ۲۵۰ حاصل می‌شود.

۲.۲.۱ پیش پردازش

ابتدا با استفاده از تابع `generate_movies` به تعداد ۳۰۰ دیتا با ۲۱ فریم ایجاد می‌کنیم. با توجه به اینکه هدف ما پیش‌بینی فریم بعدی بر اساس فریم‌ها تا لحظه‌ی کنونی است، برچسب دیتاها از شیفت فریم‌های تصویر به اندازه‌ی یک فریم حاصل می‌شود که این کار با استفاده از تابع `create_shifted_frames` انجام می‌شود.

سپس ۱۰ درصد از این دیتا را به منظور test انتخاب می‌کنیم و باقی را به تابع fit می‌دهیم. در تابع fit ۱۰ درصد از دیتای باقی‌مانده را به منظور validation در هر ایپاک استفاده می‌کنیم و باقی دیتا به منظور آموزش شبکه استفاده می‌شود.

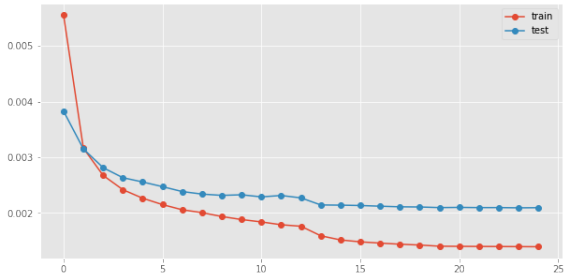
۳.۱ سلول‌های مختلف بازگشتی

در این بخش می‌خواهیم با استفاده از ساختار شبکه‌ای که در بخش قبل به آن پرداخته شد، عملکرد لایه‌های بازگشتی متفاوت را بررسی کنیم و بهترین نوع را برای شبکه نهایی خود انتخاب کنیم. برای این منظور از سه نوع لایه‌ی LSTM ، GRU و RNN استفاده می‌کنیم و این شبکه‌ها را از نظر تعداد پارامتر، سرعت آموزش و پیش‌بینی و همچنین عملکرد پیش‌بینی برای ۵ دیتای تست بررسی می‌کنیم. روند بررسی به این صورت است که به هر یک از شبکه‌ها ۷ فریم ابتدایی را می‌دهیم و شبکه نهایتا باید فریم هشتم هر ویدیو را پیش‌بینی کند. در هر یک از نتایج، فریم پیش‌بینی شده، فریم اصلی و تفاضل این دو فریم را نمایش می‌دهیم.

همچنین روند آموزش و نمودار خطای مدل را برای هر یک از این ۳ نوع لایه بازگشتی بررسی می‌کنیم و نهایتا شبکه‌ای را بر می‌گزینیم که عملکرد مناسبی بر روی دیتای validation داشته باشد.

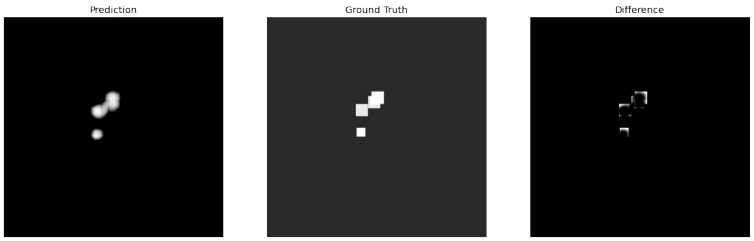
LSTM ۱.۳.۱

نمودار خطای شبکه برای دیتای تست و آموزشی به صورت زیر است:

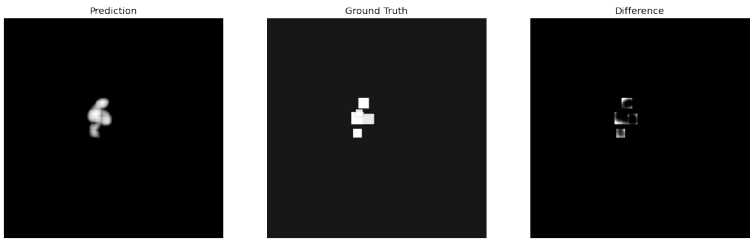


شکل ۲: نمودار خطا

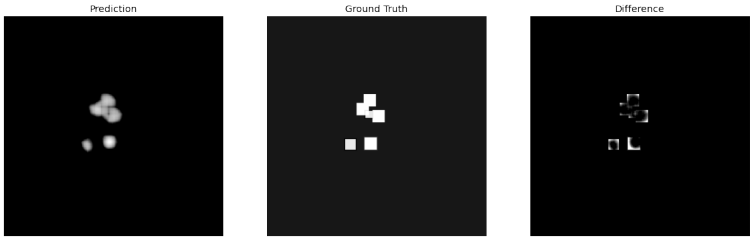
همانطور که مشاهده می‌شود شبکه عملکرد مناسبی را داشته است و خطای دیتای تست نیز روند مشابهی را همانند دیتای آموزشی داشته‌است. اکنون پیش‌بینی مدل را برای ۵ ویدیو متفاوت از دیتای تست بررسی می‌کنیم.



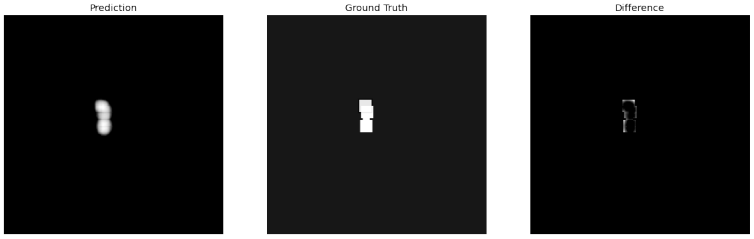
شکل ۳: پیش‌بینی اول



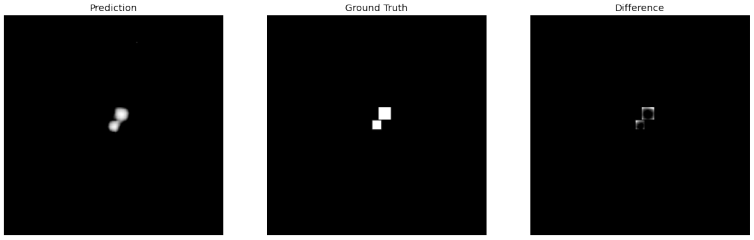
شکل ۴: پیش‌بینی دوم



شکل ۵: پیش‌بینی سوم



شکل ۶: پیش‌بینی چهارم



شکل ۷: پیش‌بینی پنجم

همانطور که مشاهده می‌شود شبکه عملکرد تقریبا مناسبی را در پیش‌بینی هشتمین فریم داشته است. خطای پیش‌بینی در تصویر سمت راست نیز نشان‌دهنده این موضوع است که شبکه تا حد معقولی توانسته فریم را پیش‌بینی کند هرچند شبکه در بازسازی لبه‌های مربع عملکرد نامناسبی داشته است و در بعضی مواقع به جای بازسازی یک مربع کامل، تنها یک توده‌ی سفید رنگ ایجاد شده‌است.

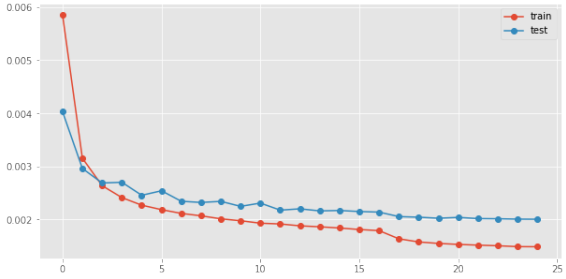
این شبکه دارای 18484730 پارامتر است که بخش خیلی زیادی از این پارامترها (تقریبا ۹۹ درصد) متعلق به لایه بازگشتی است. این شبکه نسبت به دو شبکه **GRU** و **RNN** تعداد پارامترهای خیلی بیشتری دارد و از این رو پیچیدگی آن نیز بیشتر است. زمان آموزش و پیش‌بینی این مدل به صورت زیر است:

۱. زمان آموزش: ۴ دقیقه و ۲۴ ثانیه

۲. زمان پیش‌بینی برای ۵ دیتا: ۱ ثانیه و ۷۴۰ میلی‌ثانیه

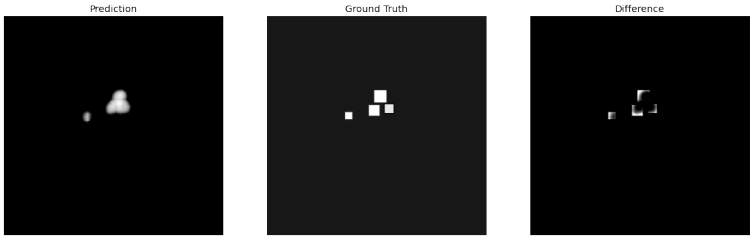
۲.۳.۱ GRU

حال در این بخش از لایه بازگشتی GRU استفاده می‌کنیم. روند آموزش و نمودار خطای شبکه در نمودار زیر نشان داده شده‌است. روند همگرایی شبکه و کاهش خطای اندکی کندتر از شبکه با لایه‌ی LSTM است، اما در کل خطای شبکه برای دیتای تست فاصله کم‌تری تا خطای داده‌ی آموزشی داشته است.

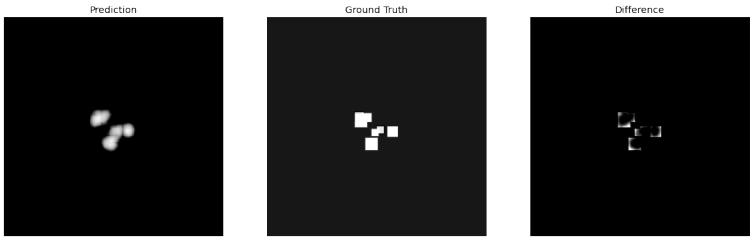


شکل ۸: نمودار خطا

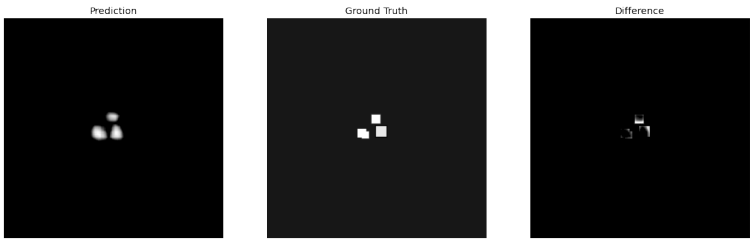
پیش‌بینی مدل برای ۵ ویدیوی متفاوت به صورت زیر است، کیفیت پیش‌بینی تقریبا همانند مدل قبلی است اما در بعضی موارد، ضعف‌های جدی‌تری در بازسازی مربع‌ها دیده می‌شود و شبکه مخصوصا در بازسازی مربع‌های کوچک‌تر یا نزدیک به هم عملکرد ضعیف‌تری را دارد.



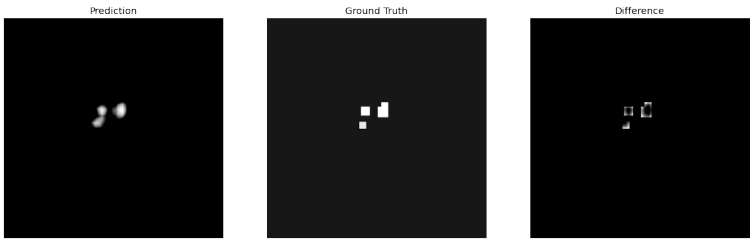
شکل ۹: پیش‌بینی اول



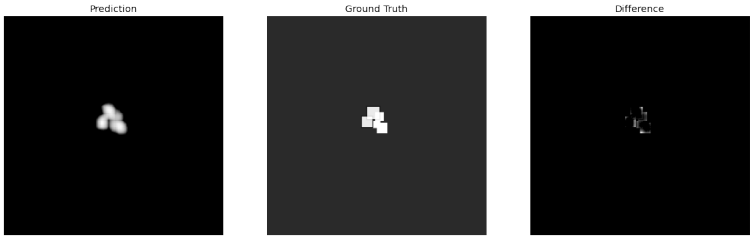
شکل ۱۰: پیش‌بینی دوم



شکل ۱۱: پیش‌بینی سوم



شکل ۱۲: پیش‌بینی چهارم



شکل ۱۳: پیش‌بینی پنجم

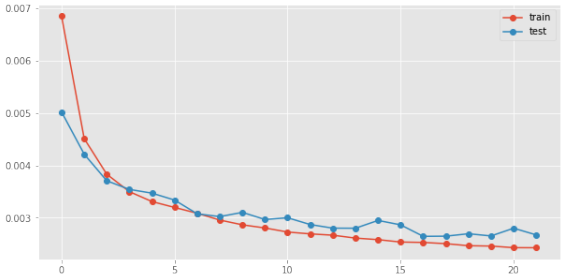
این شبکه دارای 13869047 پارامتر است که از این تعداد ۹۹ درصد متعلق به لایه بازگشتی است. این شبکه با تقریبا ۵ میلیون پارامتر کم‌تر نسبت به شبکه‌ی LSTM توانسته است به دقت مشابهی برسد و از این رو می‌تواند گزینه‌ی مناسبی باشد. اما با بررسی زمان آموزش و پیش‌بینی در هر دو مدل در می‌یابیم که تقریبا شبکه‌ی LSTM عملکرد بهتری داشته است.

۱. زمان آموزش: ۴ دقیقه و ۲۴ ثانیه

۲. زمان پیش‌بینی برای ۵ دیتا: ۲ ثانیه و ۸۰ میلی‌ثانیه

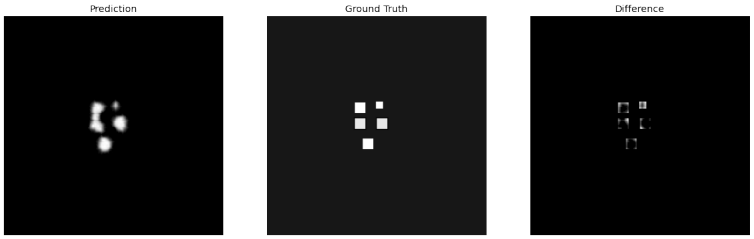
RNN ۳.۳.۱

حال در این بخش از لایه بازگشتی RNN استفاده می‌کنیم. روند آموزش و نمودار خطای شبکه در نمودار زیر نشان داده شده‌است. در این مورد خطای شبکه برای دیتای تست و آموزشی خیلی نزدیک است اما شبکه توانایی کم‌تری برای کاهش خطا داشته است در حالی که دو شبکه قبل در مدت زمان کم‌تری به خطای 0.002 رسیده‌اند.



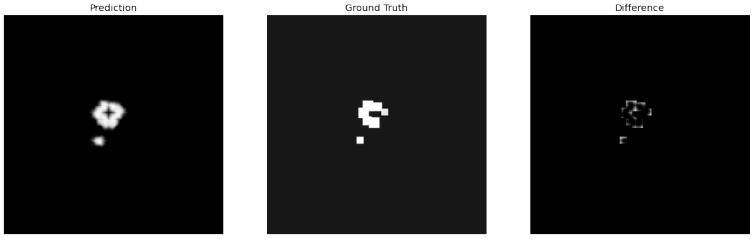
شکل ۱۴: نمودار خطا

پیش‌بینی مدل برای ۵ دیتای تست متفاوت به صورت زیر است. همانطور که مشاهده می‌شود عملکرد مدل نسبت به دو مدل قبل خیلی ضعیف‌تر است و مخصوصا برای مربع‌های کوچک‌تر خطای زیادی در تصاویر سمت راست دیده می‌شود که موید این موضوع است که شبکه نتوانسته است بازسازی مناسبی را از فریم هشتم داشته باشد.

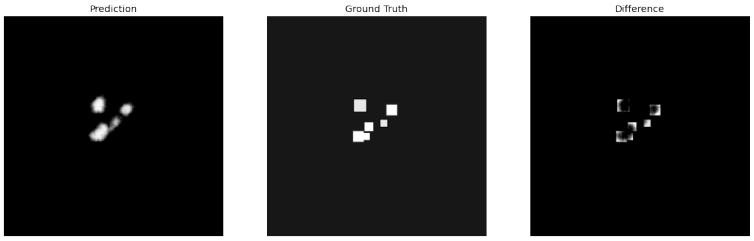


شکل ۱۵: پیش‌بینی اول

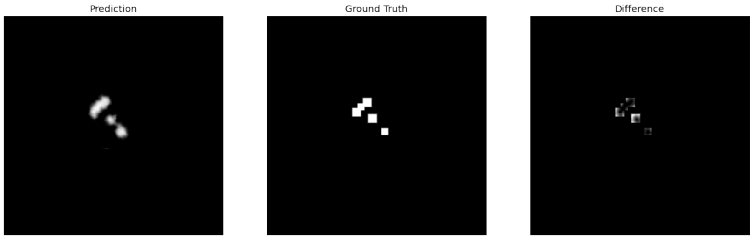
در تصویر فوق، مربعی که در بالا سمت راست تصویر است، خیلی ضعیف در تصویر بازسازی شده ایجاد شده‌است و از این رو خطای مربوط به این مربع زیاد است.



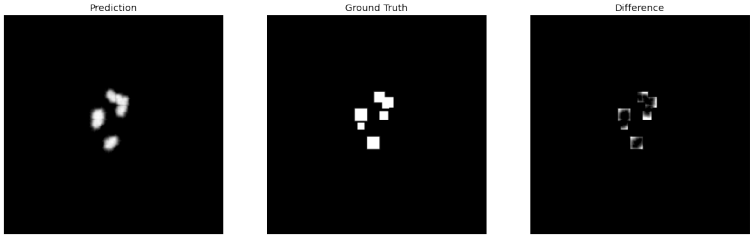
شکل ۱۶: پیش‌بینی دوم



شکل ۱۷: پیش‌بینی سوم



شکل ۱۸: پیش‌بینی چهارم



شکل ۱۹: پیش‌بینی پنجم

این شبکه تنها با دارا بودن 4629032 پارامتر، کم‌ترین تعداد پارامتر را بین این سه شبکه دارد و از این رو دارای کم‌تری پیچیدگی است. اما با بررسی زمان آموزش و پیش‌بینی مدل در می‌یابیم که عملکرد بهتری را نسبت به دو مدل قبلی از خود نشان نداده است و از این رو با توجه به پیش‌بینی ضعیفی که داشته است، این مدل را رد می‌کنیم.

۱. زمان آموزش: ۴ دقیقه و ۲۸ ثانیه

۲. زمان پیش‌بینی برای ۵ دیتا:۱ ثانیه و ۷۰۰ میلی‌ثانیه

٤.١ دو تابع خطای متفاوت

حال در این بخش می‌خواهیم عملکرد شبکه با لایه‌ی LSTM را با استفاده از دو تابع خطای متفاوت بررسی کنیم. هدف این شبکه آن است تا بر اساس ۷ فریم ابتدایی یک ویدیو، فریم هشتم آن را ایجاد کند، پس در این صورت خطای شبکه به میزان تفاوت تصویر ایجاد شده و تصویر اصلی هشتمین فریم بستگی دارد. برای محاسبه‌ی اختلاف دو تصویر می‌توانیم

از دو تابع خطای MSE و Binary Cross Entropy استفاده کنیم. این دو تابع، خطا را به ازای هر پیکسل محاسبه می‌کنند و فاصله هر پیکسل از تصویر ایجاد شده را تا تصویر اصلی به ما می‌دهند.

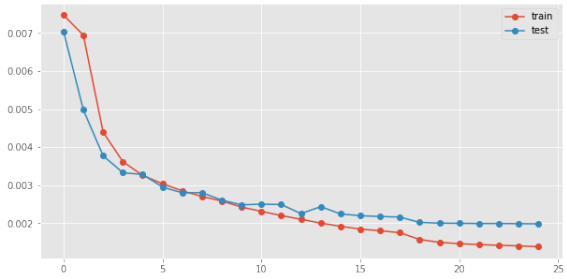
MSE ۱.٤.۱

رابطه‌ی این تابع خطا به صورت زیر است:

$$\text{MSE} = \frac{1}{N}\sum_{i}(y_i - \hat{y}_i)^2$$

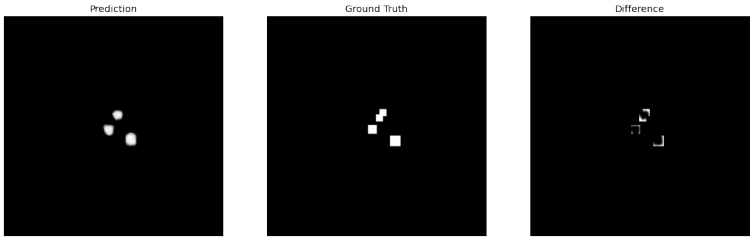
بر اساس این رابطه‌ی فاصله هر پیکسل را از دو تصویر محاسبه می‌کنیم این فاصله بر مبنای فاصله اقلیدسی تعریف می‌شود.

عملکرد شبکه به ازای استفاده از این تابع خطا به صورت زیر است:



شکل ۲۰: نمودار خطا

پیش‌بینی مدل برای یک دیتای تست با استفاده از این تابع خطا به صورت زیر است:



شکل ۲۱: پیش‌بینی

همانطور که مشاهده می‌شود شبکه توانایی خیلی خوبی در بازسازی پیکسل‌های سفید رنگ در تصویر نهایی دارد، زیرا شبکه به دنبال کاهش فاصله‌ی اقلیدسی بین دو تصویر است، از این رو تلاش می‌کند تا با ایجاد نواحی سفید رنگ در محل مربع‌ها، این فاصله را کاهش دهد ولی همزمان این موضوع باعث می‌شود تا شبکه نتواند بازسازی مناسبی از گوشه‌ی مربع‌ها داشته باشد، زیرا بازسازی این لبه‌ها نیازمند آن است که شبکه خطا را خیلی کم‌تر کند. همین موضوع باعث می‌شود تا مربع‌هایی که همپوشانی دارند به خوبی پیش‌بینی نشوند.

یک راهکار برای این موضوع که در ادامه بررسی شده‌است، استفاده از مدل با توانایی بالاتر در بازسازی تصویر است. راهکار دوم این است که از یک تابع خطای وزن‌دار استفاده کنیم تا به خطای مربوط به لبه‌های هر مربع حساس‌تر باشد.

از این رو شبکه تنها به ایجاد هاله‌ای برای هر مربع راضی نمی‌شود زیرا اگر لبه‌ها را به درستی ایجاد نکند جریمه خواهد شد و با خطای بیشتری مواجه می‌شود. این روش قابل پیاده‌سازی است اما به دلیل مدت زمان کمی که برای انجام این پروژه داشتیم آن را انجام ندادیم.

۲.٤.۱ Binary Cross Entropy

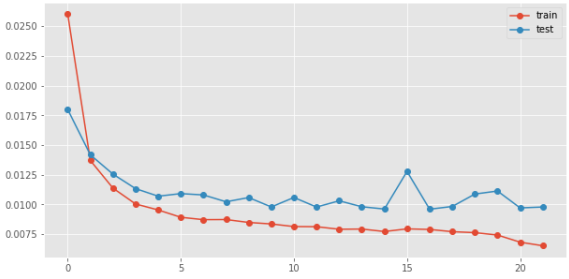
این تابع خطا به صورت زیر است:

$$H = -\frac{1}{N}\sum_{i} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)$$

این تابع خطا یک تعبیر احتمالاتی دارد. مساله‌ی مورد نظر در این بخش را می‌توان به صورت دو کلاس سفید و سیاه بررسی کرد. به این صورت که هر پیکسل یا سفید است یا سیاه. از این رو این تابع خطا به این صورت تعبیر می شود که احتمال سفید بودن یا سفید نبودن را می‌تواند بیان کند و مثلا خطا به خاطر سفید نبودن پیکسل‌های مربوط به مربع‌ها یا سفید بودن برای پس‌زمینه ایجاد می‌کند.

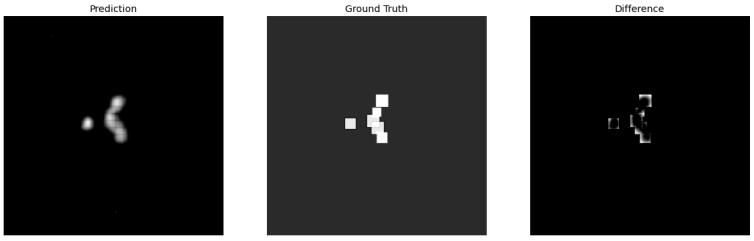
مطابق با رابطه‌ی این خطا، اگر پیکسل سفید باشد، مقدار $\log(\hat{y}_i)$ به خطا اضافه می‌شود، پس مدل باید احتمال بالاتری را برای سفید بودن به این پیکسل اختصاص دهد.

همچنین اگر پیکسل سیاه باشد، مقدار $\log(1 - \hat{y}_i)$ به خطا اضافه می‌شود از این رو مدل باید احتمال کم‌تری را برای سفید بودن به این پیکسل اختصاص دهد.



شکل ۲۲: نمودار خطا

عملکرد مدل و مقدار خطا در این شبکه در نمودار فوق نشان داده شده‌است. همانطور که مشاهده می‌شود، مدل عملکرد تقریبا ضعیف‌تری را بر روی دیتای تست داشته‌است.



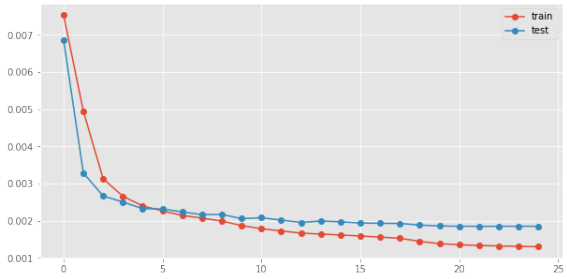
شکل ۲۳: پیش‌بینی

مطابق با تصویر فوق، شبکه در این حالت پیش‌بینی ضعیف‌تری را داشته‌استو هرچند که مربع‌ها به درستی بازسازی شده‌اند ولی علاوه بر ضعف بازسازی لبه‌ها، ضعف در بازسازی مراکز مربع‌ها نیز دیده می‌شود که موجب زیاد بودن فاصله‌ی دو تصویر می‌شود. در نتیجه در ادامه، از تابع خطای MSE استفاده می‌کنیم.

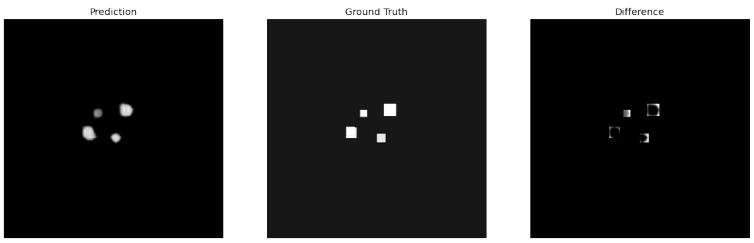
۵.۱ دو تابع بهینه‌ساز متفاوت

حال با استفاده از دو روش بهینه‌سازی متفاوت، عملکرد شبکه را بررسی می‌کنیم. نمودار خطا و همچنین یک پیش‌بینی برای هر مدل نیز ارائه شده‌است.

Adam ۱.۵.۱

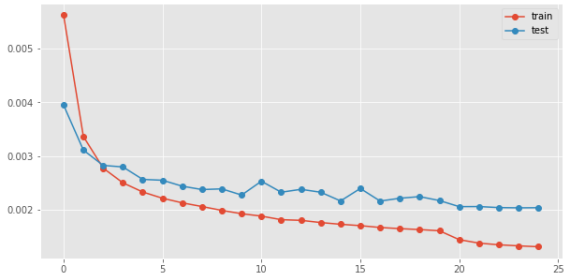


شکل ۲۴: نمودار خطا

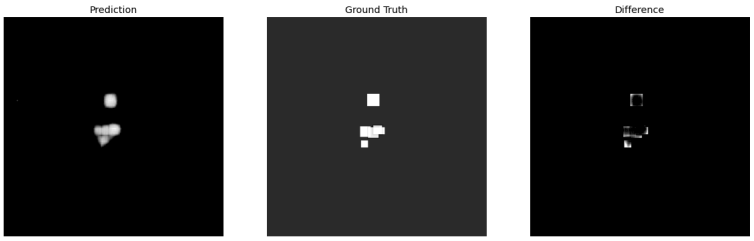


شکل ۲۵: پیش‌بینی

RMSProp ۲.۵.۱



شکل ۲۶: نمودار خطا

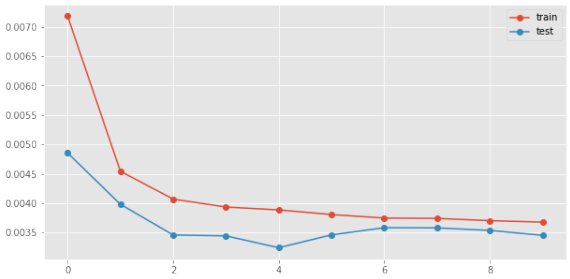


شکل ۲۷: پیش‌بینی

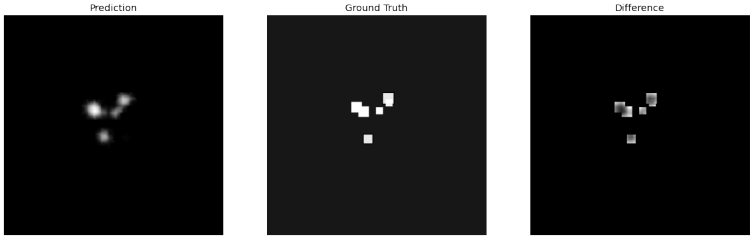
همانطور که مشاهده می‌شود، بهینه‌ساز Adam همگرایی سریع‌تری را داشته است و در این حالت خطای تست و آموزشی، اختلاف خیلی کم‌تری را دارند.

Dropout ٦.١

در این بخش به قسمت‌های مختلف شبکه، لایه‌ی dropout اضافه می‌کنیم و عملکرد آن را بررسی می‌کنیم. برای مثال نتایج زیر مربوط به زمانی است که پس از دو لایه اول یک dropout با احتمال ۳۰ درصد و پس از لایه بازگشتی یک dropout با احتمال ۲۰ درصد و پس از لایه چهارم و پنجم، dropout با احتمال ۱۰ درصد قرار دهیم. با تغییر تعداد این لایه‌های dropout و مکان قرارگیری آن‌ها و حتی استفاده از لایه Batch Normalization پیش از dropout در یافتیم که عملکرد شبکه ضعیف‌تر می‌شود. از این رو در شبکه نهایی از این لایه استفاده نمی‌کنیم.



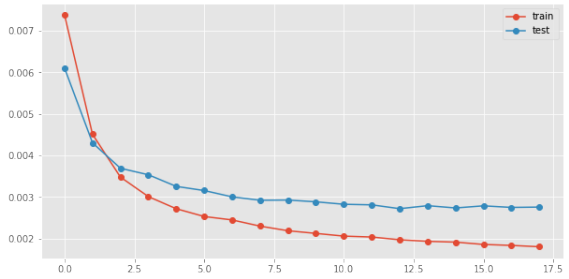
شکل ۲۸: نمودار خطا



شکل ۲۹: پیش‌بینی

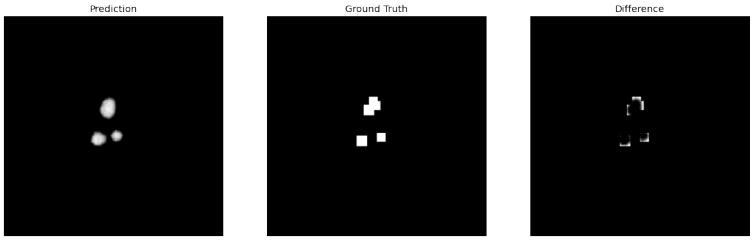
۷.۱ لایه‌های بازگشتی بیشتر

در این بخش یک لایه بازگشتی دیگر با تعداد ۹۶۱ واحد را به شبکه اضافه می‌کنیم و در این صورت ۲ لایه بازگشتی متوالی خواهیم داشت. شبکه در این حالت ۲۶ میلیون پارامتر خواهد داشت. نمودار خطا به صورت زیر است:



شکل ۳۰: نمودار خطا

طبق این نمودار افزودن این لایه عملکرد مدل را بر روی دیتای تست ضعیف‌تر کرده‌است و خطای مدل در این حالت افزایش یافته است. بنابراین در مدل نهایی تنها از یک لایه بازگشتی استفاده می‌کنیم زیرا این پیچیدگی بیشتر در مدل، با عملکرد بهتر همراه نشده است.

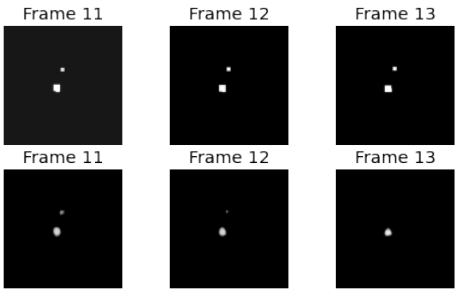


شکل ۳۱: پیش‌بینی

۸.۱ پیش‌بینی فریم‌های آینده

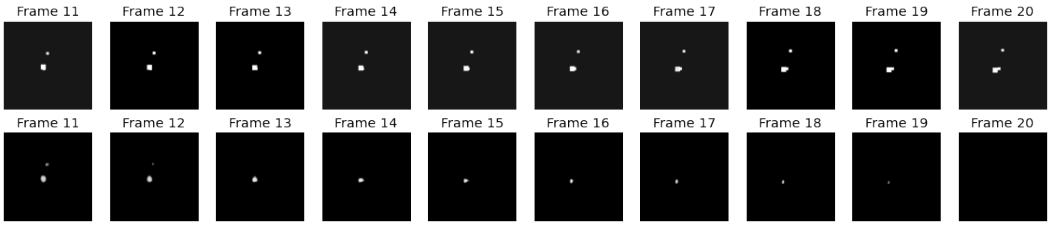
در این بخش از شبکه‌ای با تابع خطای MSE ، بهینه‌ساز adam ،یک لایه بازگشتی و بدون استفاده از dropout استفاده می‌کنیم.

در ابتدا ۱۰ فریم ابتدایی هر ویدیو را به شبکه می‌دهیم و ۳ فریم بعدی را پیش‌بینی می‌کنیم. به این صورت که پس از پیش‌بینی یازدهمین فریم، فریم یازدهم را به همراه ۱۰ فریم ابتدایی به شبکه می‌دهیم و فریم دوازدهم را پیش‌بینی می‌کنیم و به همین ترتیب.



شکل ۳۲: پیش‌بینی ۳ فریم بعدی ویدیو

در این تصویر، سطر اول تصاویر واقعی هر فریم و سطر دوم پیش‌بینی مدل را نشان می‌دهد.



شکل ۳۳: پیش‌بینی ۱۰ فریم بعدی ویدیو

۹.۱ مدل کانولوشنی بازگشتی

در بخش‌های قبلی، روشی را در پیش گرفتیم که ابتدا با استفاده از یک شبکه عصبی بیانی را برای تصاویر ایجاد کنیم و سپس بر اساس این بیان ایجاد شده، از شبکه بازگشتی بهره بگیریم. در نهایت حالت درونی لایه بازگشتی را مجددا به تصویر تبدیل کردیم. همانطور که مشاهده شد نتیجه کار چندان قابل قبول نیست. مشکل اصلی از آنجاست که تبدیل تصویر به یک بردار و عبور آن از لایه بازگشتی، رابطه‌ی فضایی بین ویژگی‌های مختلف تصویر را نادیده می‌گیرد. از این رو شبکه نمی‌تواند به دقتی که مطلوب باشد دست یابد.

در این بخش ما از لایه کانولوشنی بازگشتی استفاده می‌کنیم که در این لایه به جای لایه‌ی fully connected از لایه‌های کانولوشنی بهره گرفته شده‌است.

۱.۹.۱ ساختار مدل

ساختار مدل استفاده شده در این بخش به صورت زیر است:

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None, 250, 250, 1 0
_________________________________________________________________
time_distributed (TimeDistri (None, None, 125, 125, 32 320
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 125, 125, 32 128
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 125, 125, 32 0
_________________________________________________________________
time_distributed_3 (TimeDist (None, None, 62, 62, 64)  18496
_________________________________________________________________
time_distributed_4 (TimeDist (None, None, 62, 62, 64)  256
_________________________________________________________________
time_distributed_5 (TimeDist (None, None, 62, 62, 64)  0
_________________________________________________________________
time_distributed_6 (TimeDist (None, None, 31, 31, 128) 73856
_________________________________________________________________
time_distributed_7 (TimeDist (None, None, 31, 31, 128) 512
_________________________________________________________________
time_distributed_8 (TimeDist (None, None, 31, 31, 128) 0
_________________________________________________________________
conv_lst_m2d (ConvLSTM2D)    (None, None, 31, 31, 128) 1180160
_________________________________________________________________
time_distributed_9 (TimeDist (None, None, 62, 62, 128) 147584
_________________________________________________________________
time_distributed_10 (TimeDis (None, None, 62, 62, 128) 0
_________________________________________________________________
time_distributed_11 (TimeDis (None, None, 125, 125, 64 73792
_________________________________________________________________
time_distributed_12 (TimeDis (None, None, 125, 125, 64 0
_________________________________________________________________
time_distributed_13 (TimeDis (None, None, 250, 250, 32 18464
_________________________________________________________________
time_distributed_14 (TimeDis (None, None, 250, 250, 32 0
_________________________________________________________________
conv3d_transpose (Conv3DTran (None, None, 250, 250, 1) 289
=================================================================
Total params: 1,513,857
Trainable params: 1,513,409
Non-trainable params: 448
_________________________________________________________________
```

شکل ۳۴: ساختار شبکه

لایه‌های شبکه عبارتند از :

۱. یک لایه کانولوشنی دو بعدی با ۳۲ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه batch norm و dropout با احتمال 0.4 اضافه شده‌است.

۲. یک لایه کانولوشنی دو بعدی با ۶۴ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه batch norm و dropout با احتمال 0.4 اضافه شده‌است.

۳. یک لایه کانولوشنی دو بعدی با ۱۲۸ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه batch norm و dropout با احتمال 0.4 اضافه شده‌است.

۴. یک لایه کانولوشنی بازگشتی با ۱۲۸ فیلتر و کرنل ۳ در ۳ که ابعاد ورودی را حفظ می‌کند.

۵. یک لایه کانولوشنی دو بعدی معکوس با ۱۲۸ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه dropout با احتمال 0.2 اضافه شده‌است.

۶. یک لایه کانولوشنی دو بعدی معکوس با ۶۴ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه dropout با احتمال 0.2 اضافه شده‌است.

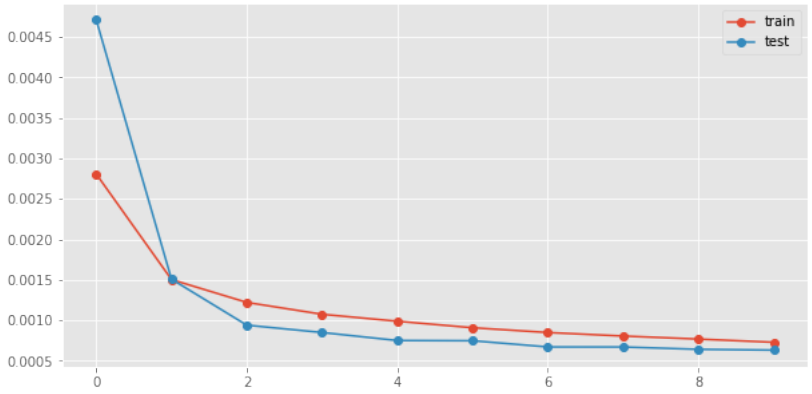
۷. یک لایه کانولوشنی دو بعدی معکوس با ۳۲ فیلتر و کرنل ۳ در ۳ و اندازه‌ی استراید ۲ که پس از آن یک لایه dropout با احتمال 0.2 اضافه شده‌است.

۸. یک لایه کانولوشنی سه بعدی معکوس با ۱ فیلتر و کرنل ۱ در ۳ در ۳ که ابعاد ورودی را حفظ می‌کند.

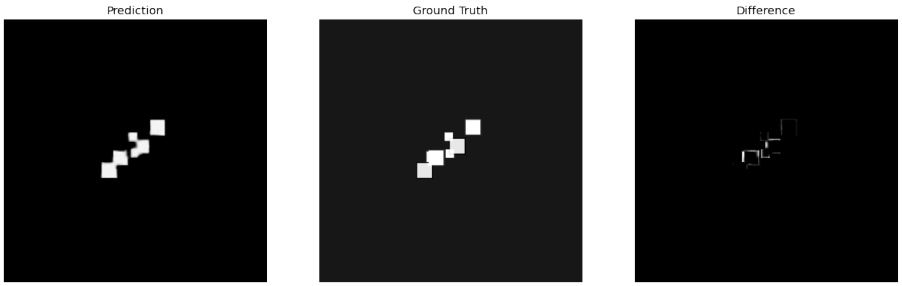
لایه‌های اول تا سوم، وظیفه‌ی استخراج ویژگی و کاهش بعد تصویر ورودی را بر عهده دارند. با هر بار نصف کردن ابعاد فضایی تصویر در این لایه‌ها، تعداد کانال‌های ویژگی دو برابر می‌شود. نهایتا feature map ایجاد شده با ۱۲۸ کانال به لایه کانولوشنی بازگشتی داده می‌شود تا بر اساس ویژگی‌های استخراج شده از لایه‌های کانولوشنی پیشین، یک حالت دو بعدی درونی را ایجاد کند. نهایتا با استفاده از لایه‌های پنجم تا هفتم، تصویر افزایش بعد داده می‌شود و در هر مرحله تعداد کانال‌های ویژگی نصف می‌گردد.

در لایه خروجی با تجمیع تصاویر ایجاد شده ناشی از فریم‌های مختلف، خروجی ۲۵۰ در ۲۵۰ پیکسل ایجاد می‌شود که پیش‌بینی مدل از فریم بعدی ویدیو می‌باشد.

نتیجه آموزش مدل به صورت زیر است:



شکل ۳۵: خطای مدل



شکل ۳۶: پیش بینی مدل از فریم هشتم بر اساس هفت فریم اول

۲ سوال سوم

در این تمرین می‌خواهیم طبقه‌بند احساسات افراد را بر مبنای توییت‌های منتشر شده طراحی کنیم. برای این منظور از لایه‌های بازگشتی در این شبکه استفاده خواهیم کرد. دیتای ورودی این شبکه، متن خام توییت و خروجی آن دو کلاس مثبت یا منفی است. بنابراین برای اعمال این ورودی به شبکه باید پیش‌پردازش‌هایی را بر روی دیتا انجام دهیم و آن را به یک دیتای عددی تبدیل کنیم.

۱.۲ نامتوازن بودن دیتا

بحث نامتوازن بودن دیتا، یکی از موارد جدی در طراحی یک طبقه‌بند است. برای مثال در یک مساله دو کلاسه، اگر دیتاهای یک کلاس نسبت به کلاس دیگر خیلی زیاد باشد، طبقه‌بند نسبت به کلاس با تعداد دیتای بیشتر بایاس می‌شود و در بعضی موارد تمایل بیشتری به کلاسی پیدا می‌کند که دیتای آموزشی بیشتری از آن دیده باشد.

در این مساله ۲۲۳۶ دیتای کلاس مثبت و ۸۴۹۳ دیتای کلاس منفی داریم ، از این رو مساله عدم توازن دیتاست را باید به گونه‌ای برطرف کنیم. روش‌های مختلفی برای مواجهه با این مساله وجود دارد. یکی از راه‌ها این است که تعداد دیتاهای کلاس بزرگ‌تر را کاهش دهیم و تعداد دیتا را در هر دو کلاس یکسان کنیم. این مورد زمانی کاربرد دارد که حجم دیتاست برای کلاس کوچک‌تر به اندازه‌ی کافی باشد و کوچک‌تر کردن دیتاست لطمه‌ای به روند آموزش شبکه وارد نکند.

روش دوم آن است که با استفاده از روش‌های مختلفی، دیتاهایی را مشابه با دیتاهای کلاس کوچک‌تر ایجاد کنیم و از این رو تعداد دیتاهای کلاس کوچک‌تر را افزایش دهیم و دیتاست را متوازن کنیم. روش‌های مختلفی برای ایجاد دیتای جدید وجود دارد که اکثر آن‌ها بر اساس روش نزدیک‌ترین همسایه و خوشه‌بندی کار می‌کنند. یکی از این روش‌ها SMOTE نام دارد.

روش سوم برای متوازن کردن دیتاست، استفاده از روش‌های augmentation مبتنی بر متن هستند. در این مورد ما از کتابخانه nlpaug و روش استفاده از کلمات مترادف استفاده می‌کنیم. به این صورت که برای هر یک از جملات مربوط به کلاس مثبت، با جایگزین کردن حداکثر ۵ کلمه با کلمات مترادف، ۳ دیتای جدید ایجاد می‌کنیم تا تعداد دیتاهای این کلاس به ۸۹۴۴ دیتا برسد.

۲.۲ پیش پردازش

در پیش‌پردازش دیتای متنی یک بخش اجباری و بخش‌های متعدد اختیاری وجود دارد. بخش اجباری انکود کردن دیتای متنی به یک دیتای عددی است که قابل اعمال به شبکه عصبی باشد. برای این بخش می‌توانیم از روش‌های مختلفی استفاده کنیم. یکی از این روش‌های استفاده از شبکه‌هایی مثل BERT است. روش‌های دیگری نیز برای Tokenize کردن متون وجود دارد به این صورت که به هر یک از کلمات موجود در متن یک عدد را اختصاص می‌دهد و سپس جمله را بر اساس کلمات موجود در آن به یک بردار عددی تبدیل می‌کند.

برای این بخش از تابع `Tokenizer` استفاده می‌کنیم. سپس بردارهای حاصل را با استفاده از تابع `pad_sequences` به بردارهای ۳۵ بعدی تبدیل می‌کنیم تا طول تمامی این بردارها یکسان شود.

اما قبل از انجام این مراحل، یک سری پیش‌پردازش‌های اختیاری را می‌توان بر روی دیتای متنی انجام داد که موجب بهبود عملکرد مدل می‌شود. مراحل این پیش پردازش به صورت زیر است:

۱. حذف لینک و آدرس ایمیل موجود در متون، زیرا در طبقه‌بندی احساسات این دیتاها تاثیر قابل تشخیصی در شبکه نخواهند داشت و از این رو بهتر است که این بخش‌ها از متون حذف شوند.

۲. حذف کاراکترهای غیراسکی [1]: این کاراکترها می‌تواند شامل ایموجی یا سایر اشکالی باشد که قابل ارسال در

[1] Non ASCII Characters

توییت‌ها هستند. شاید به نظر برسد که در نظر گرفتن کاراکترهایی مثل ایموجی می‌تواند در تشخیص احساسات موثر باشد ولی تاثیر این موضوع عملا به خاطر تعداد زیاد این کاراکترها در دیتاست و زیاد شدن تعداد کلمات منحصر به فرد در دیتاست، دیده نمی‌شود و عملا تاثیر مناسبی در آموزش ندارد.

۳. حذف فاصله‌های اضافی و کاراکترهای `\n` و `\t`

۴. حذف علائم نگارشی ! ? | ,...

۵. lower case: رعایت این مورد موجب می‌شود تا تفاوتی بین کلماتی که تنها در بزرگ یا کوچک بودن حروف متفاوت هستند وجود نداشته باشد.

رعایت این موارد باعث می‌شود تا شبکه تمرکز بیشتری بر روی دیتاهایی داشته باشد که احتمالا تاثیر بیشتری در تشخیص احساسات خواهند داشت.[۲]

همچنین برچسب‌های هر دیتا را نیز باید به گونه‌ای تبدیل کنیم که در شبکه قابل استفاده باشد. برچسب هر دیتا در دیتاست اصلی به صورت Positive یا Negative است. از این رو ابتدا این برچسب‌ها را به صفر و یک تبدیل می‌کنیم و سپس با استفاده از تابع `to_categorical` آن‌ها را به بردارهای دو بعدی تبدیل می‌کنیم که با خروجی شبکه قابل مقایسه باشد.

۳.۲ طراحی و آموزش مدل

مدل طراحی شده به صورت زیر است:

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_12 (Embedding)     (None, 35, 256)           2555136
_________________________________________________________________
lstm_12 (LSTM)               (None, 128)               197120
_________________________________________________________________
dense_24 (Dense)             (None, 64)                8256
_________________________________________________________________
dense_25 (Dense)             (None, 2)                 130
=================================================================
Total params: 2,760,642
Trainable params: 2,760,642
Non-trainable params: 0
```

شکل ۳۷: ساختار مدل

در این شبکه ابتدا از یک لایه‌ی Embedding استفاده می‌کنیم. این لایه بردار ورودی که مبین یک جمله است را به دنباله‌ای از کلمات تبدیل می‌کند که به وسیله‌ی لایه‌ی بازگشتی قابل بررسی باشد. تعداد نورون‌های این لایه را برابر با ۲۵۶ انتخاب می‌کنیم. سپس در لایه بعدی از یک لایه بازگشتی استفاده می‌کنیم که تعداد نورون‌های آن برابر با ۱۲۸ انتخاب شده‌است و مقدار dropout ۵۰ درصدی برای آن انتخاب کرده‌ایم.

این لایه بازگشتی، دنباله‌ی ۳۵ عضوی کلمات را در یک جمله مورد بررسی قرار می‌دهد و این دنباله را به یک استیت درونی که یک بردار ۱۲۸ عضوی است تبدیل می‌کند. سپس این بردار ۱۲۸ عضوی را به عنوان ورودی یک طبقه‌بند fully connected با دو لایه استفاده می‌کنیم. این طبقه‌بند از دو لایه تشکیل شده‌است که لایه اول آن دارای ۶۴

[۲] ما همچنین تلاش کردیم تا کلمات stopwords را نیز از متون حذف کنیم زیرا این کلمات تاثیری در تشخیص احساسات ندارند اما با حذف این کلمات به دقت کم‌تری رسیدیم، زیرا این کلمات می‌توانند مشتمل بر افعال مثبت یا منفی نیز باشند که بر اساس آن‌ها تا حدی می‌توان احساسات افراد را متوجه شد.

نورون و تابع فعال‌ساز ReLU می‌باشد. لایه خروجی شبکه نیز دارای ۲ نورون است که بیانگر دو کلاس مثبت و منفی است و تابع فعال‌سازی آن نیز softmax انتخاب شده‌است تا خروجی این لایه بیان احتمالاتی داشته باشد.

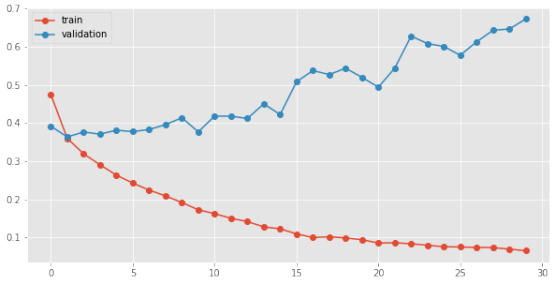
تابع خطای این شبکه `categorical_crossentropy` انتخاب شده‌است، که البته چون مساله‌ی ما یک مساله‌ی دو کلاسه است می‌توانیم از تابع خطای `binary_crossentropy` نیز استفاده کنیم. این تابع خطا یک تعبیر احتمالاتی دارد. برای مثال در این مورد با دو کلاس، یک احتمال برای کلاس مثبت تعریف می‌شود. شبکه به دنبال افزایش احتمال مثبت بودن است، در زمانی که برچسب حقیقی شبکه مثبت باشد و به دنبال کاهش احتمال مثبت بودن است زمانی که برچسب واقعی آن منفی باشد.

همچنین بهینه‌ساز adam را برای آموزش این مدل استفاده کرده‌ایم.

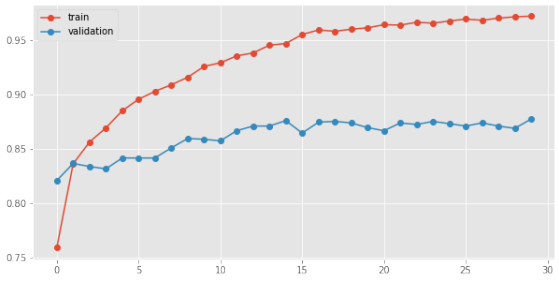
برای آموزش مدل دیتای پیش‌پردازش شده را به ۳ دسته تقسیم می‌کنیم. ابتدا ۲۰ درصد از دیتا را به تصادف انتخاب می‌کنیم و برای تست نهایی عملکرد شبکه در نظر می‌گیریم. هنگام آموزش نیز با استفاده از توابع کتابخانه keras ۱۰ درصد از دیتای باقی‌مانده را برای validation و ۷۰ درصد باقی‌مانده را برای آموزش مدل استفاده می‌کنیم.

۴.۲ تحلیل نتایج

اکنون شبکه را به ازای ۳۰ ایپاک آموزش می‌دهیم و نمودار دقت و خطای شبکه را ترسیم می‌کنیم:



شکل ۳۸: نمودار خطا

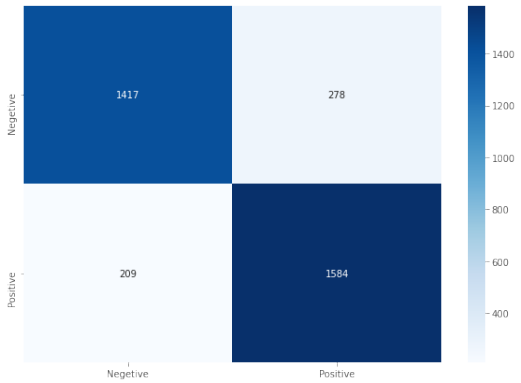


شکل ۳۹: نمودار دقت

همانطور که مشاهده می‌شود شبکه خیلی زود دچار فرابرازش می‌شود و خطای شبکه به ازای دیتای تست زیاد می‌شود در حالی که دقت آن تقریبا برابر با ۸۷ درصد باقی می‌ماند.

۵.۲ ارزیابی مدل

اکنون برای بررسی عملکرد مدل، دیتای تست را به شبکه می‌دهیم و خروجی آن را با برچسب حقیقی دیتا بررسی می‌کنیم. ماتریس آشفتگی برای دیتای تست به صورت زیر خواهد بود:



شکل ۴۰: ماتریس آشفتگی

همانطور که از ماتریس آشفتگی دریافت می‌شود، شبکه توانایی خوبی را در تشخیص دیتاهای هر دو کلاس از خود نشان داده‌است و خطای نسبتا کمی را در طبقه‌بندی این دیتاها دارد. همچنین با توجه به اینکه دیتاست را به منظور آموزش مدل، متوازن کردیم، بایاسی در پیش‌بینی مدل دیده نمی‌شود. سایر پارامترهای ارزیابی این شبکه به صورت زیر است:

۱. precision: کلاس مثبت: ۸۵ درصد. کلاس منفی ۸۷ درصد.

۲. recall: کلاس مثبت: ۸۴ درصد. کلاس منفی ۸۸ درصد.

۳. f1-score: کلاس مثبت: ۸۵ درصد. کلاس منفی ۸۷ درصد.