به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین امتیازی

نام و نام خانوادگی	عرفان باقری سولا – محمد قره حسنلو
شماره دانشجویی	۸۱۰۱۹۸۳۶۱ - ۸۱۰۱۹۸۴۶۱
تاریخ ارسال گزارش	۱۴۰۲.۲.۱۵

فهرست

جدول‌ها

No table of figures entries found.

پاسخ ۱. تخمین قیمت رمز ارزها

توضیحات مدلها

LSTM نسبت به GRU معماری پیچیده تری دارد و از memory cell و سه gate تشکیل شده است (input gate, forget gate, output gate). در مقابل GRU از دو gate (reset gate and update gate) تشکیل شده است.

شبکه‌ی (LSTM) از یک سلول حافظه برای ذخیره اطلاعاتی که به وابستگی‌های بلند مدت نیاز دارند استفاده می‌کند. دروازه فراموشی تعیین می‌کند که اطلاعاتی که در سلول حافظه وجود دارند باید نگه داشته شوند یا حذف شوند و دروازه ورود تعیین می‌کند که چه اطلاعات جدیدی باید ذخیره شوند.

شبکه‌ی (GRU) همچنین یک عنصر حافظه دارد، اما سلول حافظه جداگانه‌ای ندارد. به جای آن، از دروازه به‌روزرسانی برای تعیین میزان فراموشی اطلاعات گذشته و میزان ذخیره‌سازی اطلاعات جدید استفاده می‌کند.

(GRU) به طور کلی از لحاظ محاسباتی کم هزینه تر از (LSTM) است زیرا دارای یک معماری ساده‌تر است. (LSTM) به دلیل معماری پیچیده‌تر خود، به زمان و داده‌های آموزشی بیشتری برای همگرایی نیاز دارد. هنگامی که منابع محاسباتی محدود هستند یا مجموعه داده موجود کوچک است، اغلب ترجیح می‌دهند از (GRU) استفاده کنند زیرا با تعداد کمتری پارامتر، عملکرد مقایسه‌پذیری را ارائه می‌دهد.

در عمل، عملکرد (LSTM) و (GRU) ممکن است بسته به وظیفه و مجموعه داده مورد نظر متغیر باشد. LSTM در مواردی که حافظه بلند-مدت اهمیت دارد، مانند ترجمه زبان، عملکرد بهتری ارائه می‌دهد.

در این مقاله از هر دو معماری استفاده شده است به این صورت که دو لایه LSTM و یک لایه GRU در کنار هم و به طور موازی داده های ورودی را دریافت و پردازش می‌کنند و از یک لایه خطی رد می کنند. در نهایت یک نورون مسئول محاسبه قیمت پیش بینی شده با استفاده از اطلاعات این دو بخش است.

مجموعه دادگان و پیش پردازش آن‌ها

مجموعه دادگان مربوط به هر دو رمز ارز را از درگاه مربوطه دریافت می‌کنیم. مجموعه دادگان دریافت شده شامل داده ها تا سال ۲۰۲۳ می‌باشد. همچنین داده Price به عنوان ویژگی مد نظر انتخاب می شود.

پیش پردازش های گفته شده را نیز روی داده ها انجام می‌دهیم. طبق گفته مقاله ابتدا همه دادگان را به صورت min-max نرمالایز می‌کنیم. البته به نظرم بهتر بود ابتدا داده ها را به صورت مناسب (دنباله هایی از قیمت در ۳۰ روز + روز های مورد نیاز برای دادگان تست) در می آوردیم و بعد در طول همان دنباله دادگان را نرمالایز می کردیم تا مدل به اعداد خاص حساس نشود و به جای آن ترند کلی و نحوه حرکت قیمت را بررسی کند. به هرحال اینجا طبق مقاله پیش می رویم. با استفاده از قطعه کد زیر داده ها را به شکل مناسب تبدیل می کنیم و تقریبا از ۸۰ درصد آنها برای آموزش مدل و از ۲۰ درصد آنها برای تست مدل استفاده می کنیم. همچنین مطمئن می‌شویم که داده های مربوط به ۱۰ روز آخر در داده های تست باشند.

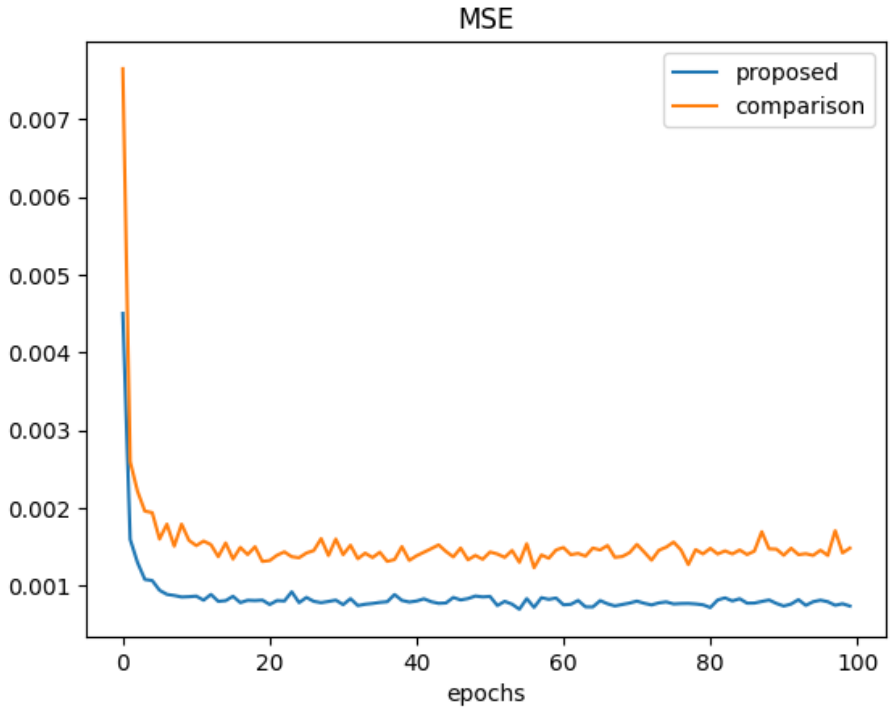
```python
def create_nn_dataset(data, w):
    data = normalizer.fit_trnsform(data)

    X = np.array([data[n: n+30] for n in range(data.shape[0]-29-w)])
    y = np.array([data[n: n+w] for n in range(30, data.shape[0]-w+1)])

    X_train, X_test, y_train, y_test = train_test_split(X[:-10], y[:-10], test_size=0.2)
    X_test = np.append(X_test, X[-10:], axis=0)
    y_test = np.append(y_test, y[-10:], axis=0)

    return X_train, X_test, y_train, y_test
```

شکل ۱ – نحوه ایجاد مجموعه داده مناسب برای آموزش مدل

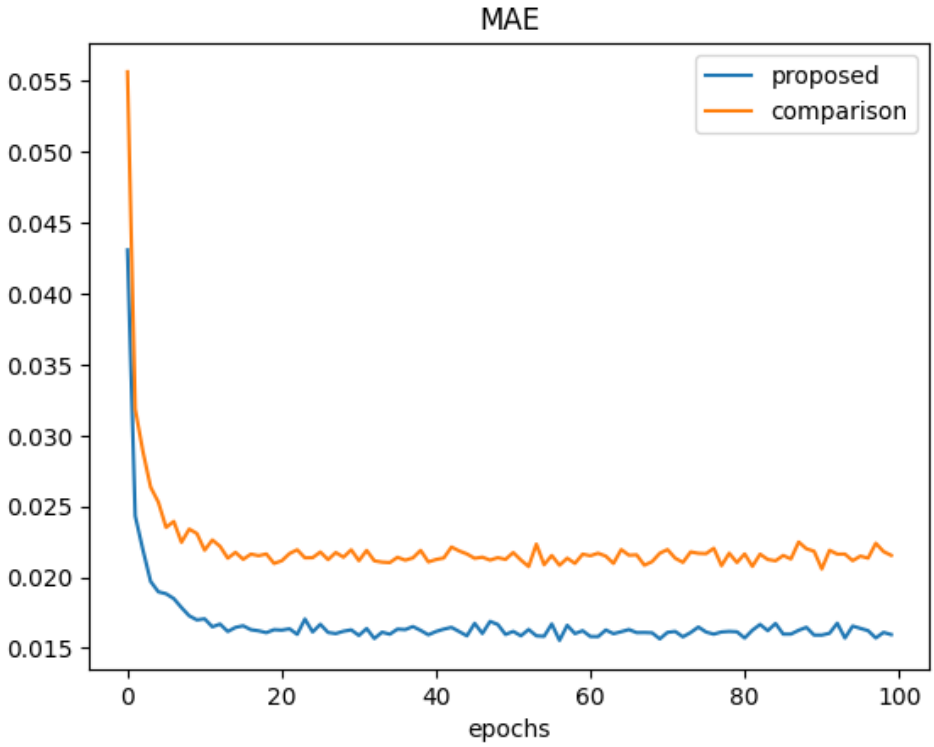
آموزش مدل‌ها

در اینجا مدل های خواسته شده را پیاده سازی می‌کنیم. در مدل پیشنهادی مقاله، تعداد نورون های هر لایه خطی که بعد از هر بخش قرار می گیرد اعلام نشده که آن را به دلخواه به صورت یک لایه با ۱۰ نورون در نظر می‌گیریم. لایه خطی نهایی نیز در هر دو مدل یک نورون دارد که با توجه به خروجی مورد انتظار انتخاب مناسبی است. همچنین احتمال DropOut نیز ذکر نشده است که در همه جا 0.5 در نظر گرفته شده است. در مدل LSTM ، بعد از لایه lstm نیز از dropout استفاده کردیم چراکه بدون استفاده از آن عملکرد مدل به شدت پایین می‌آمد.

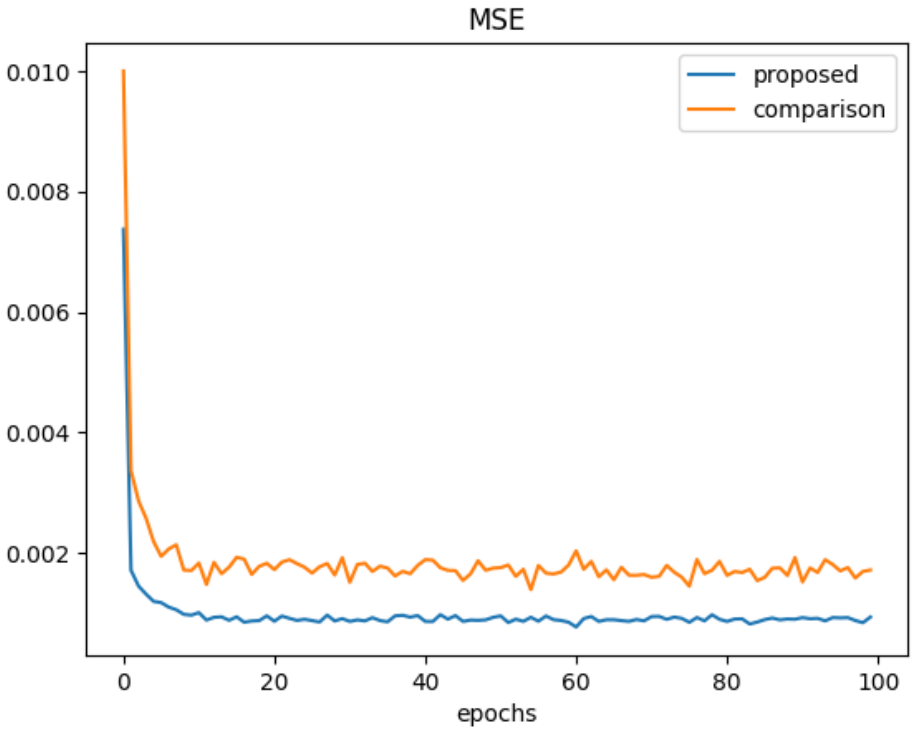
هر دو مدل را می سازیم هر دو را در 100 epoch آموزش می دهیم. نبود مجموعه داده برای validation و همچنین عدم استفاده از Early Stopping در مقاله جالب نیست و به نظرم عملکرد کلی مدل را کاهش می دهد چراکه هم از overfitting اطلاع خاصی نداریم و هم طبق نمودار هایی که در ادامه از دو متریک خواسته شده مشاهده خواهیم کرد، آموزش مدل ها تا 100 epoch واقعا توجیه خاصی ندارد و خیلی سریعتر از آن می توان آموزش را متوقف کرد.



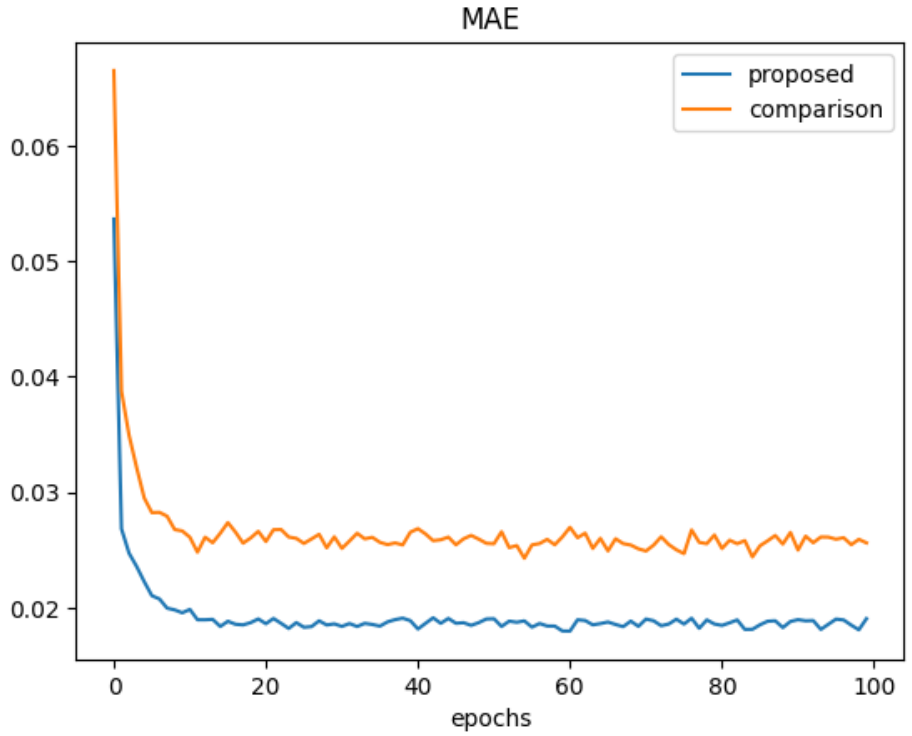
شکل ۲ – نمودار **MSE** در طول آموزش برای **Litecoin**



شکل ۳ – نمودار **MAE** در طول آموزش برای **Litecoin**



شکل ۴ – نمودار **MSE** در طول آموزش برای **Monero**



شکل ۵ – نمودار **MAE** در طول آموزش برای **Monero**

ارزیابی و تحلیل نتایج

هر دو مدل برای هر رمز ارز در یک فایل جداگانه آموزش داده شده و ارزیابی شده اند. در ادامه نتایج ارزیابی را مشاهده می کنیم. باید توجه داشت که داده های نرمالایز شده به مدل داده شده اند ولی نتایج دینرمالایز شده اند و ارزیابی روی قیمت های واقعی انجام شده است. (به همین دلیل است که به خاطر قیمت های بالاتر ارز Monero اعداد خطا در آن معمولا بزرگتر هستند)

جدول ۱ – جدول نهای برای پنجره ۱ روزه

Model	Currency	MSE	RMSE	MAE	MAPE
LSTM	Litecoin	82.33	9.07	5.37	0.06
	Monero	162.31	12.74	7.61	0.06
Proposed	Litecoin	72.62	8.52	5.04	0.06
	Monero	123.30	11.10	6.76	0.006

جدول ۲ – جدول نهایی برای پنجره ۳ روزه

Model	Currency	MSE	RMSE	MAE	MAPE
LSTM	Litecoin	144.43	12.02	6.43	0.07
	Monero	237.28	15.40	8.87	0.06
Proposed	Litecoin	146.86	12.12	6.37	0.07
	Monero	214.04	14.63	8.55	0.07

جدول ۳ – جدول نهایی برای پنجره ۷ روزه

Model	Currency	MSE	RMSE	MAE	MAPE
LSTM	Litecoin	256.08	16.00	8.68	0.09
	Monero	413.52	20.34	11.51	0.08
Proposed	Litecoin	265.06	16.28	8.71	0.09
	Monero	393.44	19.84	11.41	0.09

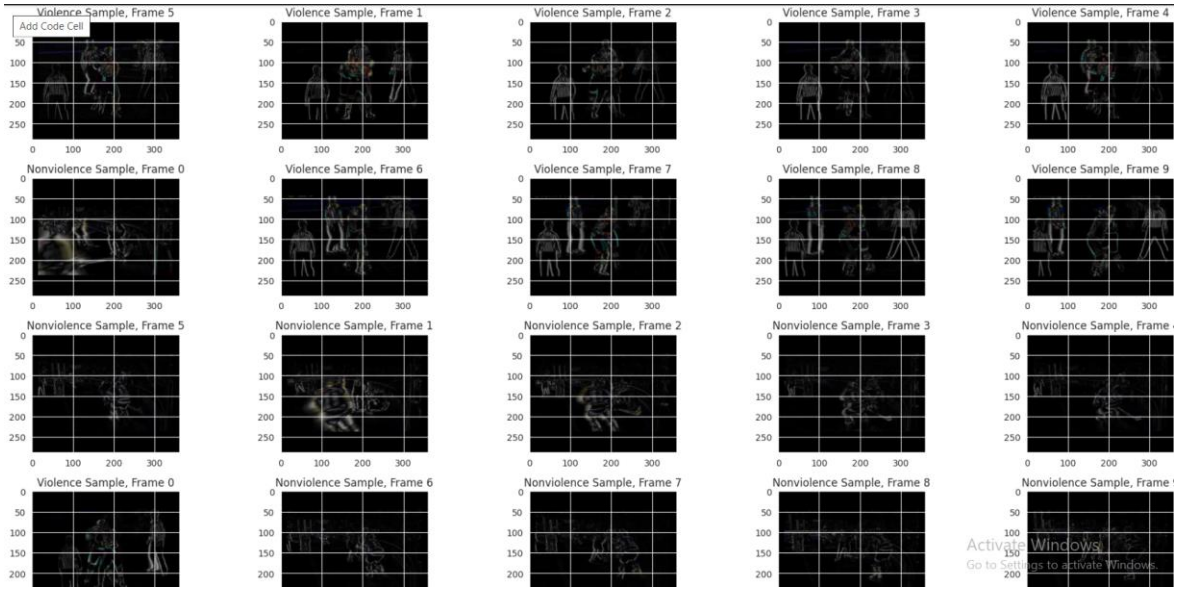
پاسخ ۲. تشخیص خشونت در فیلم

دریافت و پیش پردازش دادگان

در این پروژه در کلاس DataLoader، برای هر batch که جلو میرویم این کارها را به ترتیب انجام میدهیم:

label را برای فیلم مشخص میکنم. فیلم را لود کرده و فریم های آن را استخراج میکنیم. سپس برای تعداد فریم هایی که میخواهیم(یعنی ۱۰ تا) این کارها را به ترتیب انجام میدهیم:

دو فریم پشت سر هم استخراج شده را از هم کم میکنیم و آن را ذخیره میکنیم. دلیل این کار آن است که همچنین در این بین ۱۰ فریم از هر دو کلاس خشونت و عدم خشونت از یک فیلم را در یکی از پوشه های مشخص شده از قبل دخیره میکنیم که نتیجه به شکل زیر میشود:



شکل ۱: نمونه فریم های استخراج شده دو فیلم با خشونت و عدم خشونت

در نهایت به اندازه interval بین دو فریم skip میکنیم. کارهای انجام شده به تعداد فریم های خواسته شده به شکل زیر میباشد:

```python
# Extract frames and save as images
frames = []

for i in range(self.num_frames):
  # Find current frame
    success, frame = cap.read()
    if not success:
        break

    # Find adjacent frame
    next_success, next_frame = cap.read()
    if not next_success:
        break

    # Calculate subtraction of frame and next_frame
    sub_frame = cv2.absdiff(frame, next_frame)
    frames.append(sub_frame)

    # You can change index according to batch_size to see another example here
    if index == 0 and label == 1:
        cv2.imwrite(os.path.join(violence_sample, f'subtractional_result_{i}.jpg'), sub_frame)
    elif index == 0 and label == 0:
        cv2.imwrite(os.path.join(nonviolence_sample, f'subtractional_result_{i}.jpg'), sub_frame)

    # Skip frames
    if frame_index_interval > 2:
        for _ in range(frame_index_interval-2):
            cap.grab()
    elif frame_index_interval == 2:
        for _ in range(frame_index_interval-1):
            cap.grab()
    elif frame_index_interval <= 1:
        for _ in range(frame_index_interval):
            cap.grab()
```

شکل ۲: استخراج و تفریق فریم ها

برای پیش پردازش های گفته شده در مقاله به شکل زیر عمل میکنیم:

ابتدا از هر طرف عکس به اندازه ای که سیاه میباشد، آن را crop میکنیم. سپس سایز عکس را به (256, 256, 3) تغییر میدهیم. به صورت رندوم و با احتمال ۵۰٪ به صورت horizontally و vertically عکس را flip میکنیم. در آخر عکس را با میانگین صفر و واریانس ۱ نرمالایز میکنیم. پیش پردازش های انجام شده به شکل زیر میباشد:

```
frames = np.array(frames)

# Data augmentation on each frame difference
frames_aug = np.zeros((self.num_frames, self.img_height, self.img_width, 3))
for i in range(frames.shape[0]):
    img = frames[i]

    # Crop 15% of each side of the image
    crop_size = int(0.15 * min(img.shape[:2]))
    img = img[crop_size:-crop_size, crop_size:-crop_size]

    # Resize image to (256, 256, 3)
    img = cv2.resize(img, (self.img_height, self.img_width))

    # Randomly flip horizontally and vertically
    if np.random.random() < 0.5:
        img = cv2.flip(img, 0)
    if np.random.random() < 0.5:
        img = cv2.flip(img, 1)

    # Normalize to make images centered around mean 0 and variance unity
    img = img.astype(np.float32)
    mean = np.mean(img)
    std = np.std(img)
    img -= mean
    img /= std
    frames_aug[i] = img

batch_x.append(frames_aug)
batch_y.append(label)
```

شکل ۳: پیش پردازش ها انجام شده

پیاده سازی مدل و آموزش

مقدار اندازه ورودی برابر (256, 256, 3) میباشد. پیاده سازی مدل به این صورت است که از وزن های pretrain شده ResNet50 استفاده میکنیم. بعد از آن یک لایه convLSTM2D با ۲۵۶ فیلتر و کرنل سایز ۳ استفاده میکنیم که باعث میشود وابستگی تغییرات با اکشن های قبلی را یاد بگیرد. بعد از batchNormalization و flat کردن، لایه های fc که به ترتیب ۱۰۰۰، ۲۵۶، ۱۰ و در نهایت یک نورون دارند(به ترتیب با توابع فعالساز relu، relu و sigmoid هستند) را برای کلاس بندی به دو کلاس خشونت و عدم خشونت استفاده میکنیم. با توجه به مقاله از optimizer که نام آن RMSprop هست استفاده میکنیم و مقدار پارامتر اولیه نرخ یادگیری آن را ۰.۰۰۰۱ میگذاریم که با توجه به آزمایشی که انجام شد، یادگیری خیلی بهتر و smooth تری نسبت به ۰.۰۰۱ داشت که در انتهای مقاله نیز به آن اشاره شده است. باتوجه به نوع مسئله که کلاس بندی است، از binary crossentropy استفده شده است.

```python
# Define the input shape
input_shape = (num_frames, height, width, 3)

# Load the pre-trained ResNet50 model
resnet = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, input_shape=(height, width, 3))

# Define the ConvLSTM layer
convlstm = tf.keras.layers.ConvLSTM2D(filters=256, kernel_size=(3, 3), strides=1, padding='same', return_sequences=False)

# Define the fully connected layers
fc1 = tf.keras.layers.Dense(1000, activation='relu')
fc2 = tf.keras.layers.Dense(256, activation='relu')
fc3 = tf.keras.layers.Dense(10, activation='relu')
fc4 = tf.keras.layers.Dense(1, activation='sigmoid')

# Define the model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Input(shape=input_shape))
model.add(tf.keras.layers.TimeDistributed(resnet))
model.add(convlstm)
model.add(BatchNormalization())
model.add(tf.keras.layers.Flatten())
model.add(fc1)
model.add(fc2)
model.add(fc3)
model.add(fc4)

# Define the RMSprop optimizer
optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.0001)

# Compile the model
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=[accuracy, precision,
                                                                         recall, f1_score])
```

شکل ۴: پیاده سازی مدل

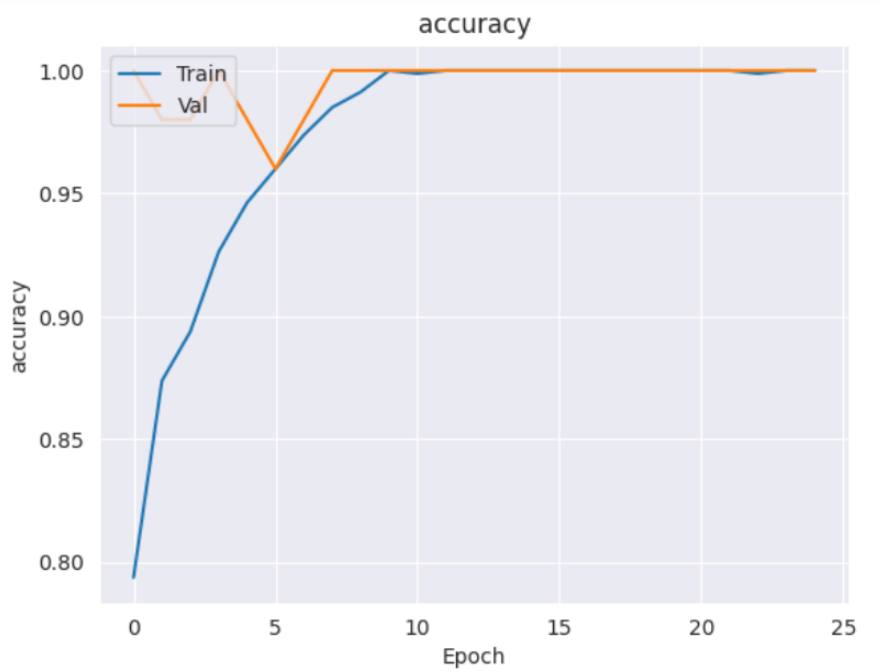
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 10, 8, 8, 2048)   23587712
 ibuted)

 conv_lstm2d (ConvLSTM2D)    (None, 8, 8, 256)         21234688

 batch_normalization (BatchN  (None, 8, 8, 256)        1024
 ormalization)

 flatten (Flatten)           (None, 16384)             0

 dense (Dense)               (None, 1000)              16385000

 dense_1 (Dense)             (None, 256)               256256

 dense_2 (Dense)             (None, 10)                2570

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 61,467,261
Trainable params: 61,413,629
Non-trainable params: 53,632
```


شکل ۵: خلاصه مدل

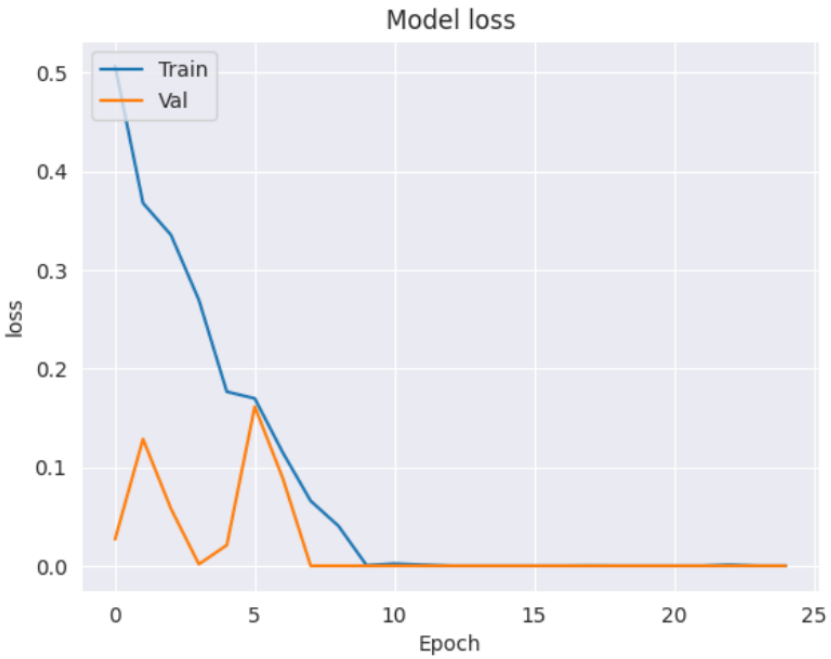
دیتا به سه دسته آموزش، اعتبارسنجی و ارزیابی شکسته است که به ترتیب درصد استفاده شده در هر کدام۰.۸، ۰.۰۵ و ۰.۱۵ میباشدهمچنین در فرایند آموزش با توجه به مقاله از EarlyStopping با پارامتر ۵ استفاده کردیم که باعث شد در ۲۵ امین epoch فرایند آموزش متوقف شود. همچنین از learning rate schedule برای داینامیک کردن فرایند یادگیری استفاده کردیم که با ۰.۰۰۰۱ شروع کرده و تا ۰.۰۰۰۰۰۳۲۵ کاهش پیدا کرده است.

نتایج

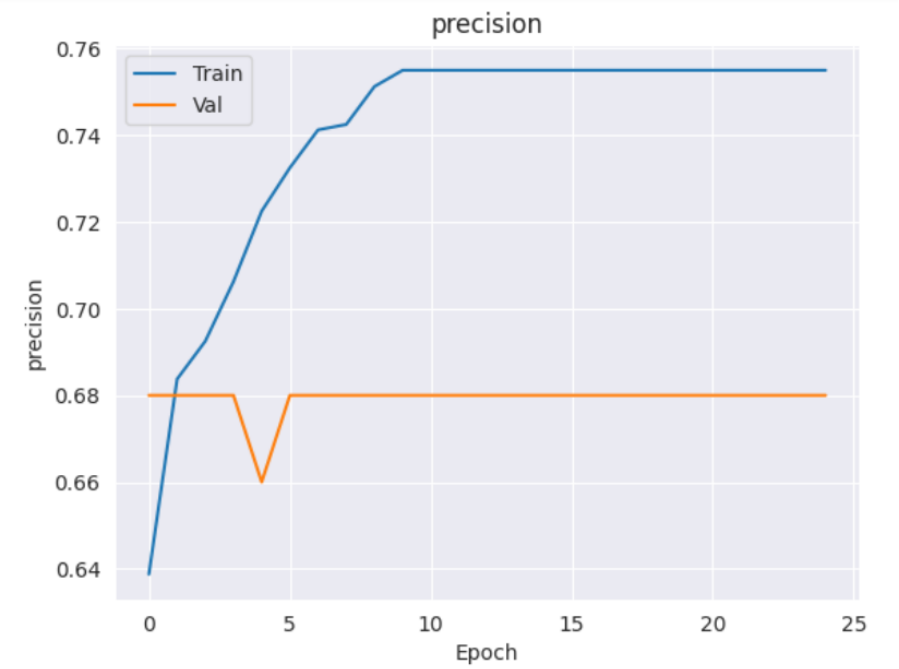
تصاویر نتایج به دست اومده به شکل زیر میباشد:



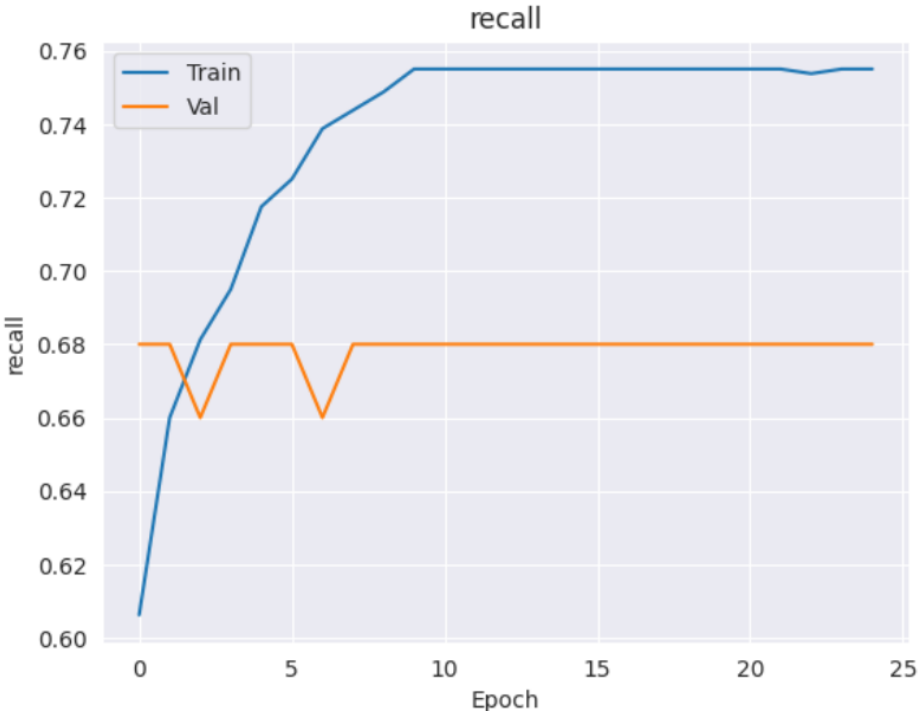
شکل ۶: نمودار دقت



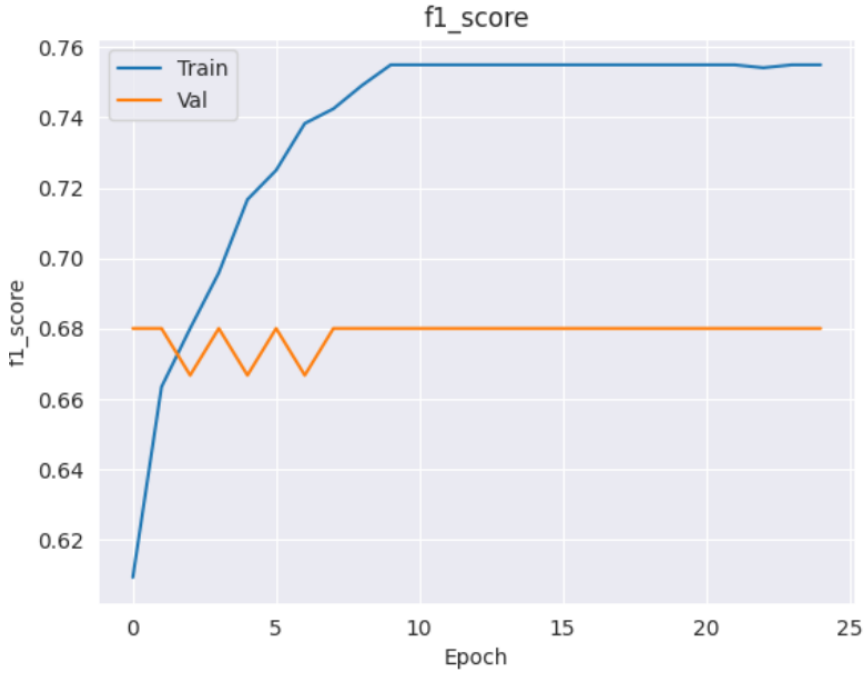
شکل ۷: نمودار **loss**



شکل ۸: نمودار **precision**



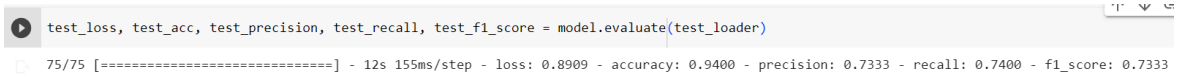
شکل ۹: نمودار **recall**



شکل ۱۰: نمودار **f1 score**

با توجه به نتایج بالا، توانسته ایم به دقت خوبی برای داده های آموزش و اعتبارسنجی برسیم. همچنین در مورد metric های دیگر در precision به ۶۸ درصد در داده های اعتبارسنجی، در recall هم به ۶۸٪ در داده های اعتبارسنجی و در نتیجه در f1_score هم به ۶۸٪ برای داده های اعتبارسنجی رسیده ایم و این smooth نبودن برای داده های validation به این دلیل است که تعداد داده ها در این دسته کم بوده(۲۵ عدد ویدئو) . آن را spark کرده است. اما در داده train چون تعداد بالا بوده و ۴۰۰ ویدئو در آن قرار دارد، به صورت smooth تری مقدار های مختلف سنجش افزایش پیدا کرده و مقدار loss کاهش پیدا میکند.

در نهایت در داده تست نتایج به شکل زیر در آمده اند:



شکل ۱۱: سنجش های مختلف در داده ارزیابی

در داده های ارزیابی که ۷۵ ویدئو در آن قرار گرفته است، دقت ۹۴٪ بوده و مقدار های دیگر اندازه گیری مانند precision، recall و f1_score برابر ۷۳٪، ۷۴٪ و ۷۳٪ میباشد که به نسبت تعداد داده های موجود به دقت خوبی رسیده ایم.

همچنین در شکل زیر ماتریس در هم ریختگی برای هر سه دسته داده آموزش، اعتبارسنجی و ارزیابی کاملا برای ما مشخص میکند که با هر ویدیویی چگونه دسته بندی انجام شده است:

```
# Compute the confusion matrices
train_cm = confusion_matrix(train_true, train_pred)
val_cm = confusion_matrix(val_true, val_pred)
test_cm = confusion_matrix(test_true, test_pred)

print('Training set confusion matrix:')
print(train_cm)
print('Validation set confusion matrix:')
print(val_cm)
print('Test set confusion matrix:')
print(test_cm)
```

```
400/400 [==============================] - 64s 156ms/step
25/25 [==============================] - 4s 164ms/step
75/75 [==============================] - 11s 150ms/step
Training set confusion matrix:
[[400   0]
 [  2 398]]
Validation set confusion matrix:
[[25  0]
 [ 0 25]]
Test set confusion matrix:
[[72  3]
 [ 5 70]]
```

شکل ۱۲: ماتریس در هم ریختگی برای هر سه دسته آموزش، اعتبارسنجی و ارزیابی

ماتریس درهم ریختگی به ما نشان میدهد که اولا تمام ویدئوها در هر سه دسته به شکل balance در دو دسته خشونت و عدم خشونت وجود دادر.(یعنی مثلا در داده تست، ۴۰۰ ویدیو با خشونت و ۴۰۰ تای دیگر بدون خشونت است.) در داده آموزش ۷۹۸ ویدئو به شکل درست بر اساس مدل به دست آمده در کلاس مدنظر قرار میگیرند و ۲ ویدئو اینگونه نیستند. در داده اعتبارسنجی همه ویدئو درست پیش بینی شده اند و در داده ارزیابی، ۱۴۲ ویدئو از ۱۵۰ ویدئو در کلاس درست و ۸ ویدئو در کلاس نادرست قرار گرفته اند.