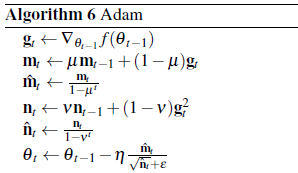
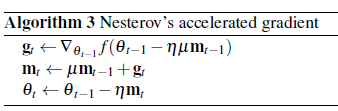
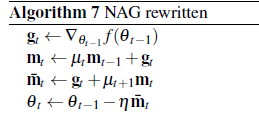
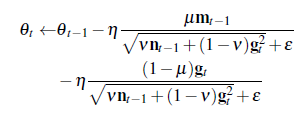
1-



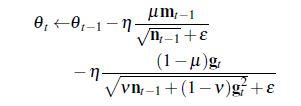
Adam، RMSProp را با momentum کلاسیک ترکیب می کند. اما نشان داده شده است که در حالت کلی NAG (Nesterov’s accelerated gradient) از momentum کلاسیک بهتر است .الگوریتم NAG را بازنویسی می کنیم تا ساده تر و کارآمدتر باشد.

بردار m̃ علاوه بر آپدیت بردار momentum برای timestep بعدی μt + 1mt ، حاوی آپدیت گرادیان (مشتق) برای timestep فعلی gt است که باید قبل از گرفتن گرادیان در timestep بعدی اعمال شود. دیگر نیازی به اعمال بردار momentum برای timestep فعلی نیستیم زیرا قبلاً آن را در آخرین آپدیت پارامترها ، در timestep t-1 اعمال کرده ایم.

قانون آپدیت Adam را می توان با توجه به بردارهای momentum/norm قبلی و آپدیت گرادیان فعلی مانند زیر نوشت.



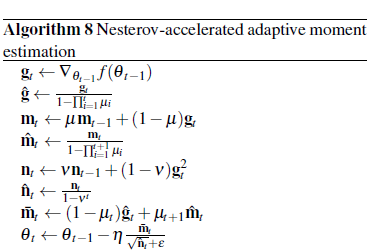
در بازنویسی NAG ، ما قسمت اول مرحله را بر می داریم و آن را قبل از گرفتن گرادیان (مشتق) تابع هزینه f اعمال می کنیم. با این حال ، مخرج به gt بستگی دارد ، بنابراین ما نمی توانیم از ترفند مورد استفاده در NAG برای این معادله استفاده کنیم . هر چند ، v به طور کلی بسیار بزرگ انتخاب می شود (به طور معمول> 90.) ، بنابراین تفاوت بین nt-1 و nt به طور کلی بسیار کم خواهد بود. پس می توانیم nt-1 را با nt بدون آنکه دقت زیادی را از دست بدهیم جایگزین می‌کنیم:



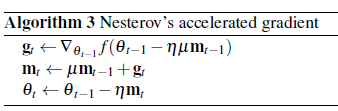
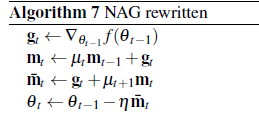
ترم اول در عبارت فوق دیگر به gt بستگی ندارد ، به این معنی که در اینجا می توانیم از ترفند Nesterov استفاده کنیم. این، عبارات زیر را برای m̃t و θt به ما می دهد:

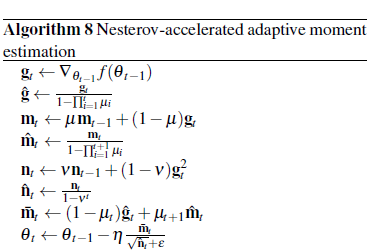
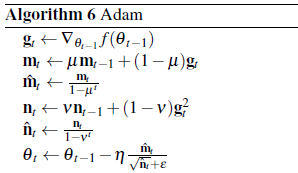


تنها چیزی که باقی مانده این است که تعیین کنیم که چگونه ترم‌های تصحیح bias مقداردهی اولیه را بگنجانیم ، با توجه به اینکه gt از timestep فعلی می آید اما mt از timestep بعدی است. این به ما فرم نهایی الگوریتم Nadam را می دهد.



خط اول کد فوق مربوط به گرادیان است. خط سوم و چهارم کد مربوط به first moment و first unbias هست. μ همان beta1 است. خط پنج و شش مربوط به second moment و second unbias می‌باشد و v همان پارامتر beta2 هست. خط آخر همان خط آخر adam هست با این تفاوت که به جای mhat از m̃t استفاده شده است. خط هفتم از الگوریتم NAG آمده است.

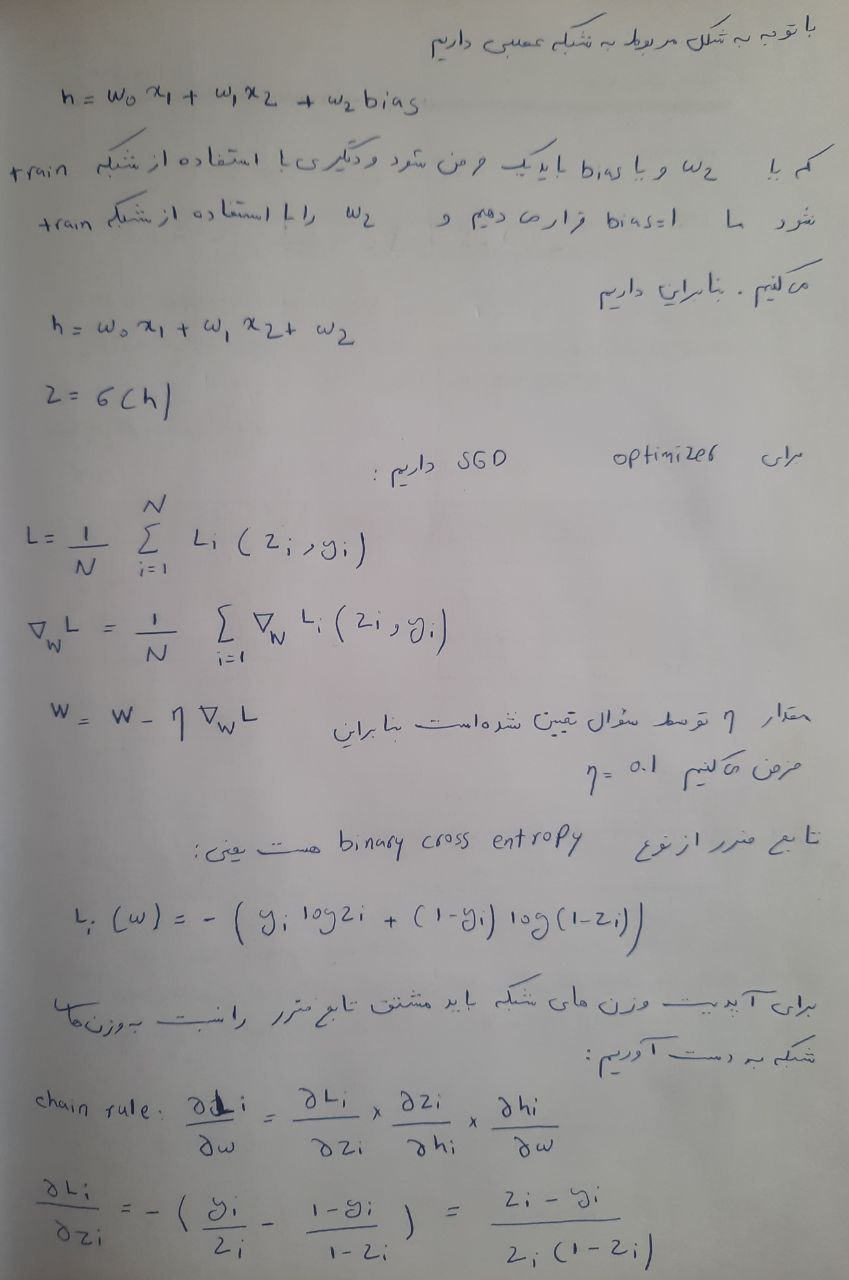


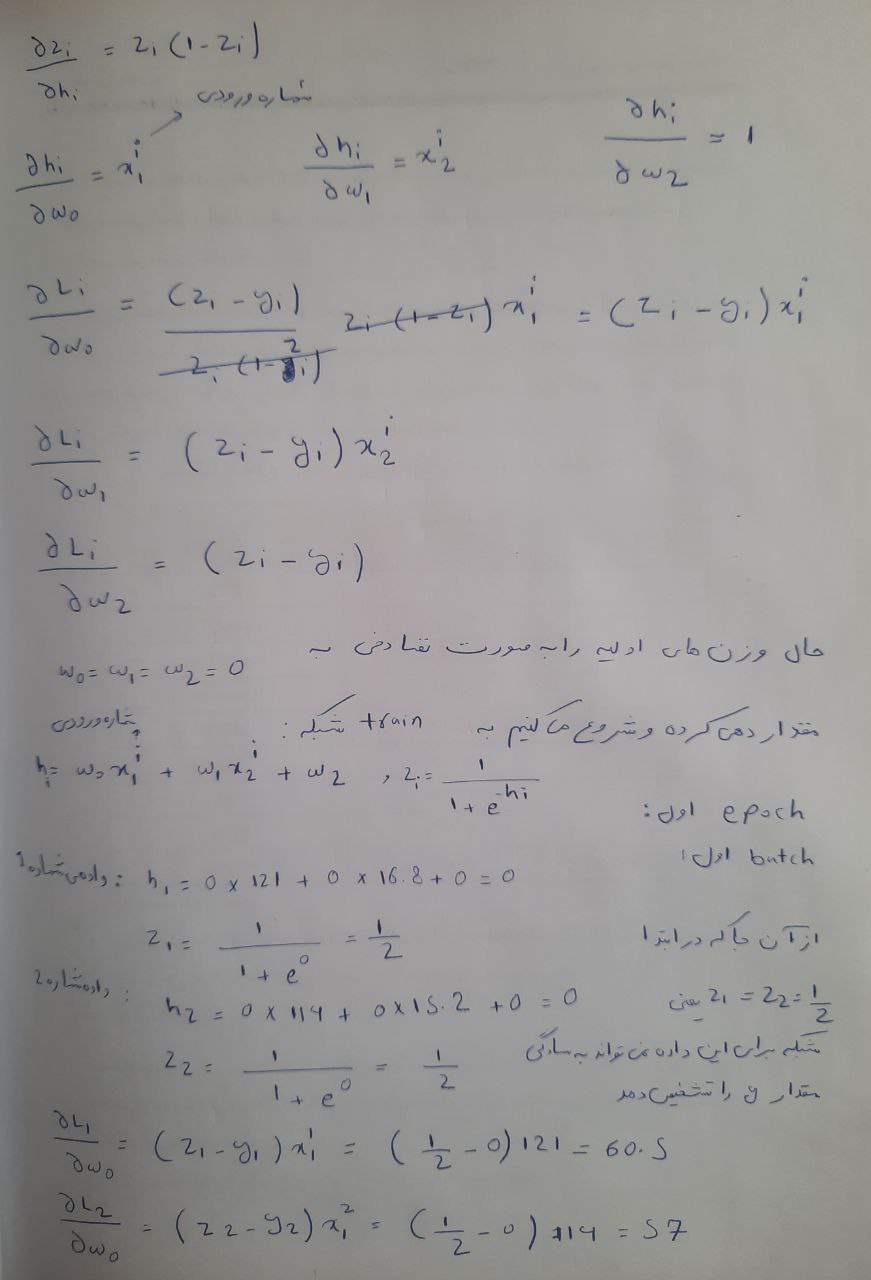


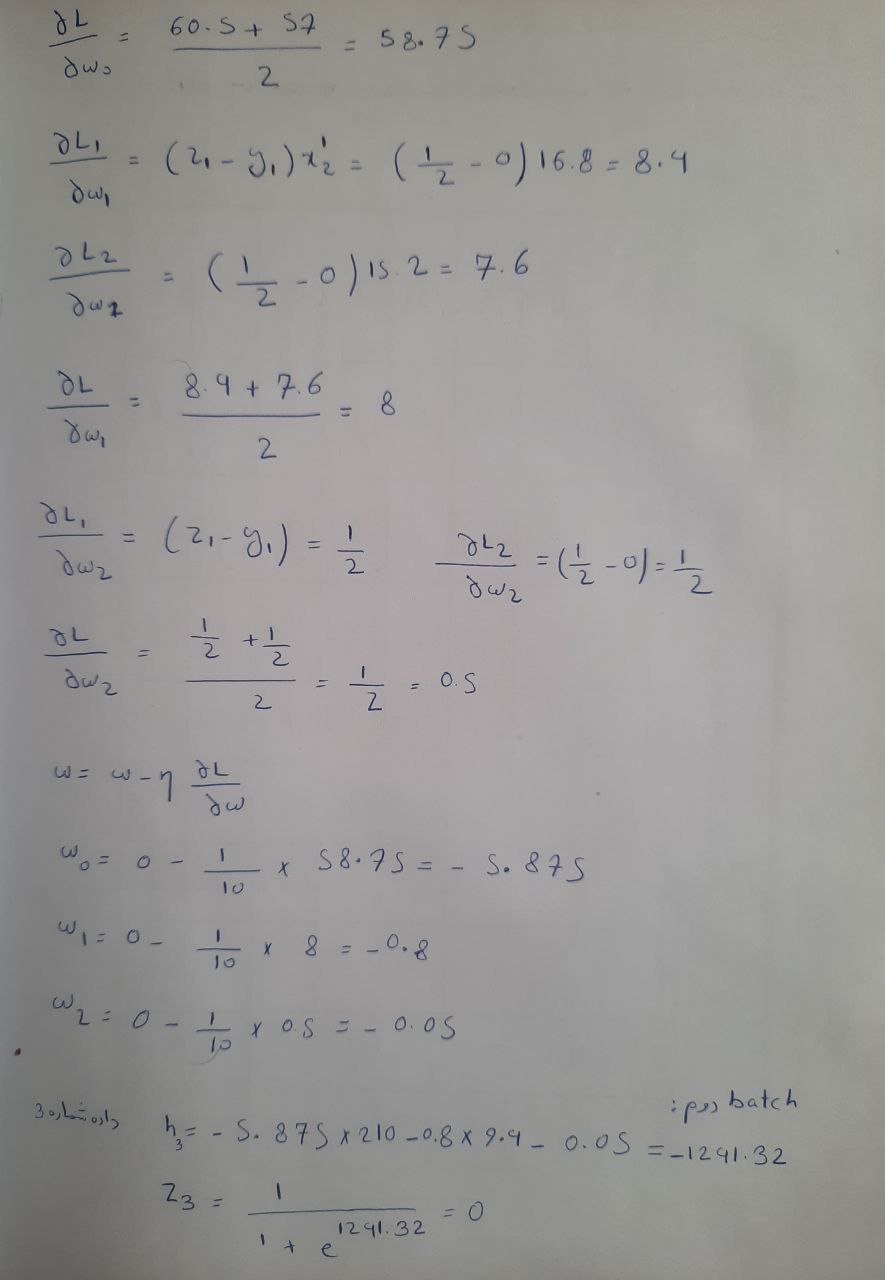
همان‌طور که باتوجه به exprimentها روی taskهای گفته شده در مقاله می‌توان فهمید loss در الگوریتم Nadam در training set و validation set کمتر از adam و performance آن بهتر می‌باشد.

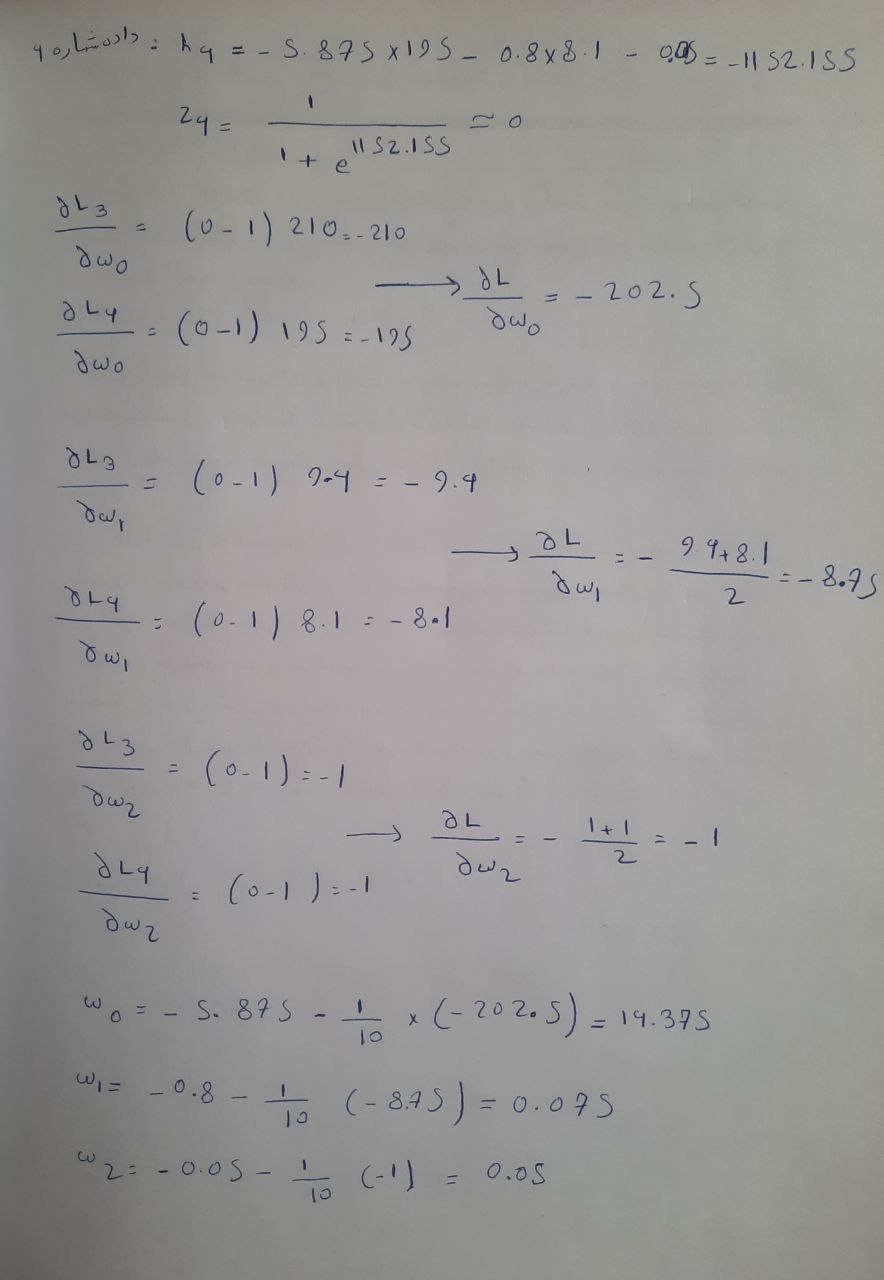
به نظر می‌آید دلیل عملکرد متفاوت optimizerها در مسئله‌های متفاوت مدل‌های متفاوت که برای حل مسائل به کار گرفته می‌شود و همچنین شکل تابع ضرر که ناشی از همین انتخاب مدل و همچنین و ویژگی‌های داده می‌باشد هست.

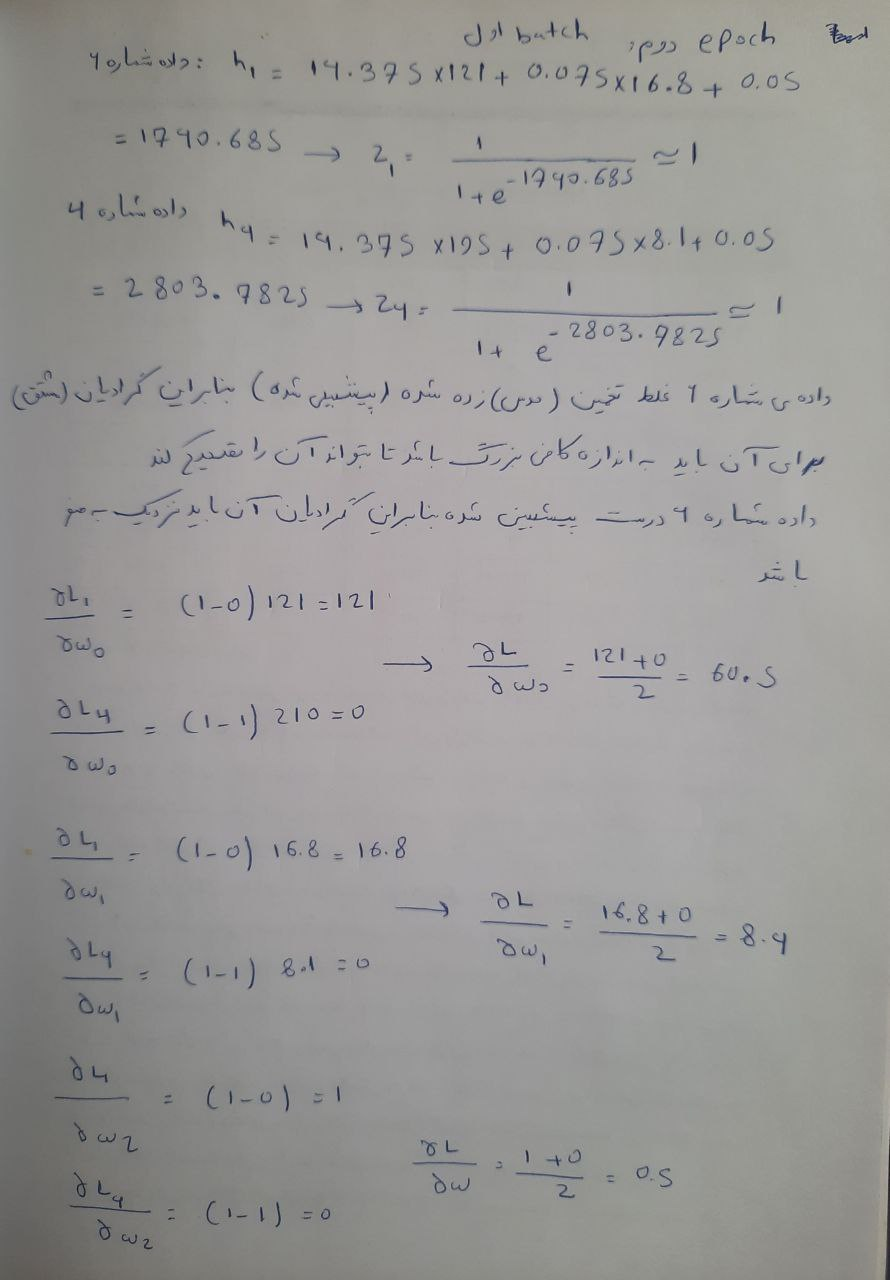
2-

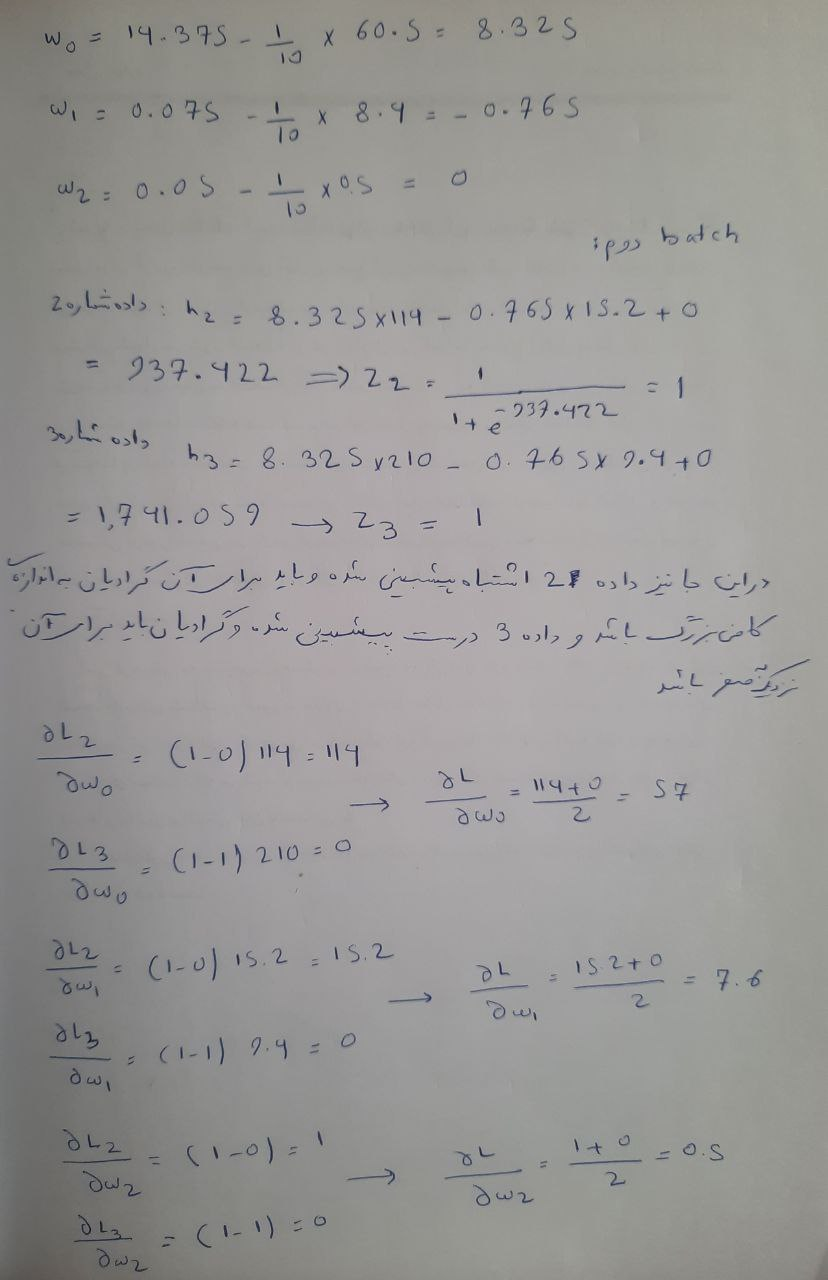


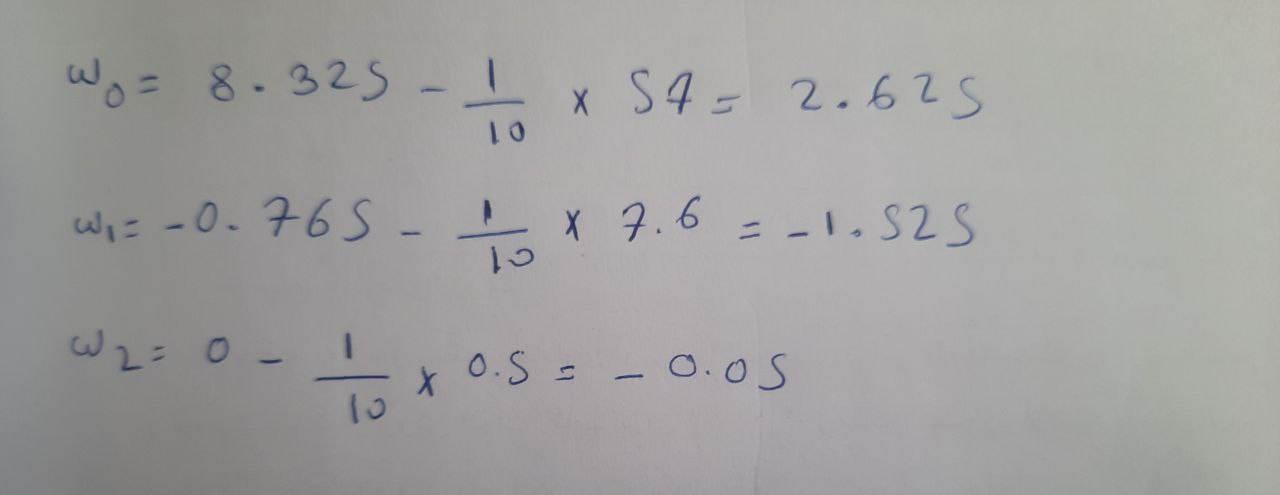












به نظر می‌رسد در این مسئله هر چه تعداد epochها بالاتر برود نتیجه بهتر می‌شود ولی شاید اگر learning rate کوچک باشد این نتیجه زودتر بهتر شود و مسئله زودتر همگرا شود چون اگر learning rate کوچک شود مقدار h کوچک می‌شود و شاید این به داده‌های یک و دو کمک کند تا زودتر همگرا شود.

https://math.stackexchange.com/questions/2503428/derivative-of-binary-cross-entropy-why-are-my-signs-not-right

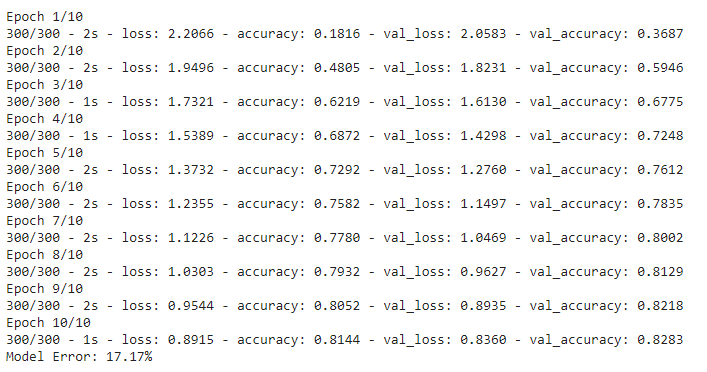
3-

امکان اینکه پارامتر‌های beta1 و beta2 طوری مقداردهی کرد که الگوریتم adam دقیقا معادل الگوریتم SGD شود وجود ندارد اما اگر beta1 = 0 و beta2 را برابر مقداری بسیار نزدیک به 1 که مساوی با آن نباشد قرار دهیم می‌توان این دو الگوریتم را به یکدیگر تقریب زد.

برای قسمت دو این سوال beta1 هر چه نزدیکتر به صفر و نه برابر با آن باشد نتیجه بهتر است و در مورد beta2 هر چه نزدیک به یک و نه برابر با آن باشد نتیجه بهتر است.

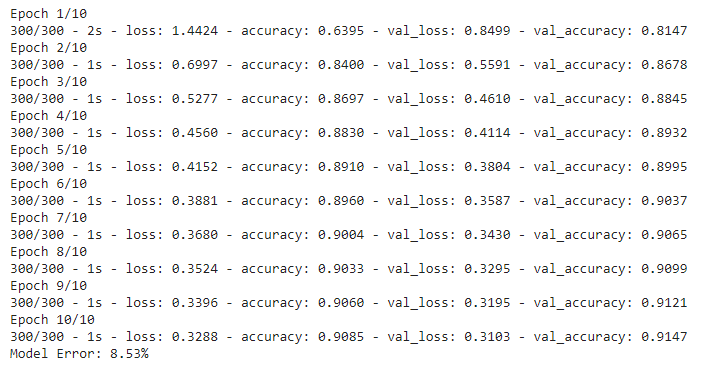
4-

Learning rate = 0.001



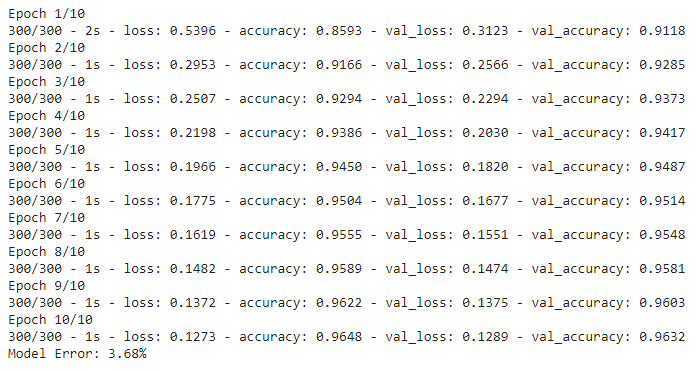
در این حالت Learning rate کوچک است بنابراین مدل دیر همگرا می‌شود و پس از 10 epoch دقت مدل به 80 رسیده که نسبت به چیزی که می‌توان به آن رسید کم می‌باشد و خطای مدل روی داده تست حدود 17 درصد می‌باشد.

Learning rate = 0.01



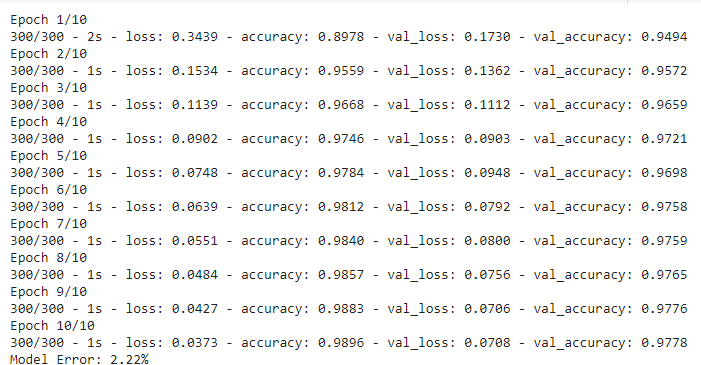
در حالت قبل پس از 10 epoch به دقت 82 رسیدیم اما با Learning rate = 0.01 پس از یکی دو epoch. بنابراین همگرایی مدل بهتر شده، یادگیری بهتر شده و خطای مدل هم به 8.5 درصد رسیده که نصف حالت قبل می‌باشد.

Learning rate = 0.1



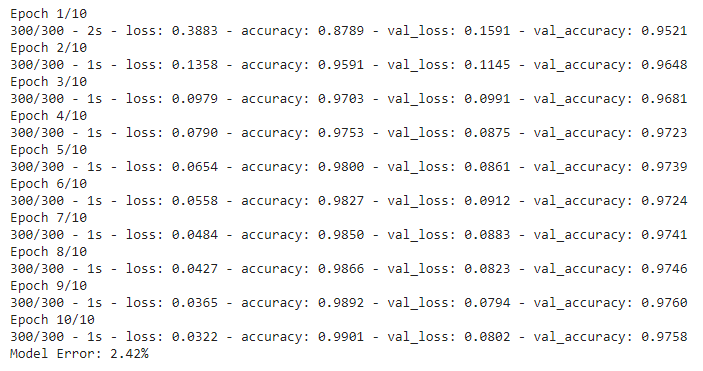
در حالت قبل پس از 10 epoch به دقت 92 رسیدیم اما با Learning rate = 0.1 پس از یکی دو epoch. بنابراین همگرایی مدل بهتر شده، یادگیری بهتر شده و خطای مدل هم به 3.68 درصد رسیده که کمتر از نصف حالت قبل می‌باشد. دقت هم به 96 درصد رسیده که دقت بالایی است و همچنین مدل هم overfit نشده است.

Learning rate = 0.5



خطای مدل هم به 2.22 درصد رسیده که بسیار کم هست. دقت هم به حدود 98 درصد رسیده که دقت بالایی است و همچنین مدل هم overfit نشده است.

Learning rate = 1



مثل حالت قبل می‌باشد و فرق چندانی نکرده است.