



دسته بندی اعداد دست نویس انگلیسی بوسیله شبکه عصبی مصنوعی

محسن رجائی

هوش مصنوعی - مهندسی کامپیوتر

دانشکده برق و کامپیوتر

[m.rajaie@ec.iut.ac.ir](mailto:m.rajaie@ec.iut.ac.ir)

## چکیده

شبکه عصبی راهکاری با کارایی فوق العاده در حوزه گسترده ای از مسائل است و این راهبر یادگیری ماشین توانایی گسترش در زمینه های مختلفی را داراست و انواع مختلفی از این شبکه هایی عصبی وجود دارند که در اینجا ما دو نوع شبکه های عصبی چند لایه پرسپترون یا MLP و شبکه عصبی کانولوشن یا CNN را مورد استفاده قرار دادیم که CNN یک نوع از شبکه های عمیق عصبی به شمار می رود. شبکه عصبی چند لایه با قدمتی بیش از ۶۰ سال هنوز هم در مسائل کلاسیک پاسخگوی ماست، اما اگر بخواهیم در کارهای واقعی از شبکه های عصبی استفاده کنیم و کارایی مطلوبی داشته باشیم باید از شبکه های عمیق که عمر آنها به حدود ۱۰ سال میرسد استفاده نماییم (منظور از ۱۰ سال این نیست که قبلاً شبکه های عمیق وجود نداشته اند، بلکه قبلاً به دلیل مشکلاتی که اینگونه شبکه ها داشتند، کنار گذاشته شده بودند و مجدداً ۱۰ سال است که مورد استفاده قرار گرفته اند و کارایی خوبی در زمینه های مختلف داشته اند)

## مقدمه

برای انجام این کار باید در ابتدا شبکه عصبی چند لایه را طراحی نماییم، ابزار مورد استفاده در این پروژه نرم افزار متلب و چارچوب (framework) MatConvNet [2] است که تمامی مراحل پیاده سازی شبکه MLP (شبکه عصبی چند لایه) توسط خودمان صورت خواهد پذیرفت یعنی ابتدا باید ساختار مورد نظر را ایجاد کنیم و سپس بعد از ایجاد شبکه با داده های آموزشی شبکه را آموزش دهیم و سپس شبکه را بهبود دهیم تا به هدف مورد نظر که همان دقت بالا بر روی داده های تست است برسیم و سپس برای پیاده سازی شبکه عصبی کانولوشن (شبکه عصبی عمیق [3] CNN) از چارچوب MatConvNet استفاده خواهیم کرد.

ابتدا ساختار شبکه ساخته شده را شرح می دهیم (این نکته قابل توجه است که پیاده سازی انجام شده به صورت پویا می باشد و تعداد لایه های شبکه و تعداد نوروں های هر لایه، تعداد ورودی ها و خروجی ها و سایر پارامترها قبل از اجرای یادگیری شبکه از کاربر در ورودی دریافت می گردد، و بوسیله این کد می توان شبکه را با هر تعداد لایه دلخواه و هر تعداد نوروں مورد نیاز ایجاد کرد (شبکه ای با عمق دلخواه ساخت)، اما در اینجا برای اینکه بتوان این شبکه را شرح داد از یک مدل خاص و خیلی ساده شده استفاده می کنیم).

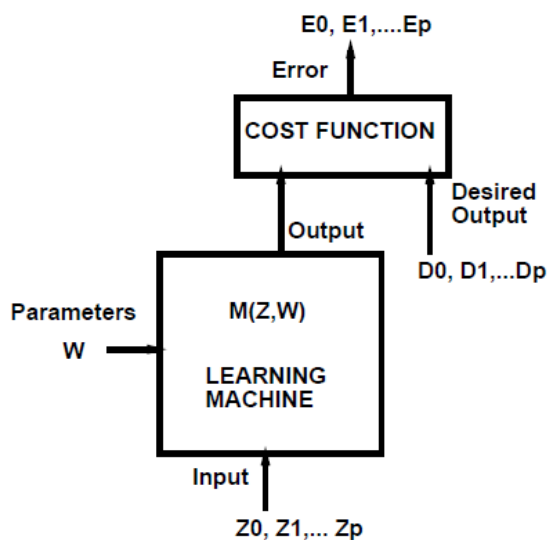


Figure 1 - Gradient-based Learning Machine

تئوری یادگیری گرادیانی با ناظر ما در شکل ۱ خلاصه شده است.

بعد از بدست آمدن مقدار خطا با روش [3] Backpropagation مقادیر  $W$  یا وزن ها را تغییر می دهیم تا مقدار خطای بدست آمده مینیمم گردد.

این یادگیری یک روش یادگیر با ناظر می باشد. و از تابع خطای مجموع مربعات استفاده می کند.

### ساختار شبکه عصبی ساده شده

چون ساختار شبکه در هنگام اجرا مشخص میشود (ورودی ها از کاربر دریافت میگردند)، و برای اینکه درک مفهوم انجام کار ساده تر باشد، ساختار و شکل زیر را به عنوان شبکه مورد استفاده در نظر بگیرید (این ساختار ساده شده ساختار اصلی است). ساختار یک شبکه ۴ لایه ای که دارای ۷۸۴ ورودی و ۱۰ خروجی و ۲ لایه مخفی می باشد مانند شکل ۲ می باشد.

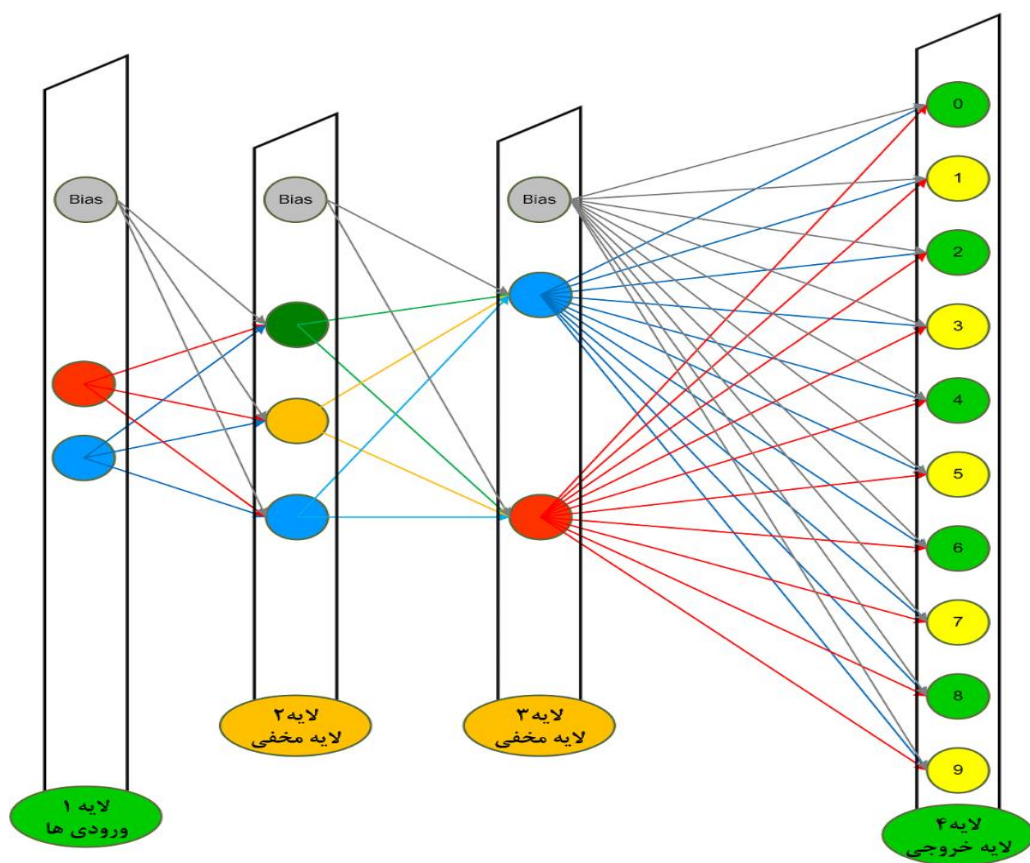


Figure 2 - MLP simple Architecture

مدل ما ۴ لایه دارد:

لایه اول ورودی ها

لایه دوم، در واقع لایه مخفی اول

لایه سوم و لایه مخفی دیگر (دوم)

لایه چهارم و در واقع لایه خروجی که ۱۰ نورون در این لایه قرار دارد.

همچنین هر لایه، بجز لایه آخر (خروجی) دارای یک نورون بایاس است.

میتوانید تصویر شبکه را در صفحه بعد ببینید.

شکل ۳ نشان دهنده ساختار ساده شده شبکه طراحی شده است:

Fields	Size	wts	z	a	delta	DW	bias	MSE	delta_bias
1	2 []	[]		2x50000 do...	[]	[]	[] []		[]
2	3 [0.0755,0.09...	[-0.0895,-0....	[0.4776,0.47...	[0,0,0]	[0,0,0;0,0,0;...		1 []		0
3	2 [0.1116,-0.0...	[0.1835,-0.0...	[0.5457,0.49...	[0,0]	[0,0;0,0;0,0;...		1 []		0
4	10 3x10 double	[0.1992,0.04...	[0.5496,0.51...	[0,0,0,0,0,0,...	3x10 double		1 10x50000 do..		0

Figure 3 - MLP simple Architecture details

ما از یک ساختار در متلب به شرح زیر برای تعریف لایه ها استفاده کردیم:

```
layer=struct('Size',[ ],'wts',[ ],'z',[ ],'a',[ ],'delta',[ ],'DW',[ ]...
,'bias',[ ],'MSE',[ ],'delta_W',[ ],'delta_bias',[ ],'big_delta',[ ],...
,'big_delta_bias',[ ],'delta_W_last',[ ]);
```

که هر لایه توسط دستور layer(i) مشخص میشود.

فیلد layer(i).Size مشخص کننده تعداد نوروں ها در لایه i ام می باشد.

فیلد layer(i).wts مشخص کننده وزن های ورودی به این لایه می باشد، در واقع وزن های ورودی از لایه i-1 به لایه i را مشخص می کند. این فیلد یک ماتریس دویبعی است که در لایه اول این ماتریس وجود خارجی ندارد (بدلیل اینکه هیچ وزنی در لایه اول که همان ورودی ها هستند معنا ندارد) و اندازه آن توسط رابطه زیر مشخص میشود:

$layer(c).wts.Size = [layer(i-1).Size+1, layer(i).Size]$

فیلد layer(i).z مشخص کننده مجموع وزن های ورودی به لایه i ضرب در ورودی های مربوط به نوروں j در لایه i می باشد.  $(z=WX)$

فیلد layer(i).a مشخص کننده اعمال تابع سیگموئید بر روی مقدار z layer(i) می باشد.  $(a=\text{sigmoid}(layer(i).z))$

فیلد layer(i).delta نیز مشخص کننده دلتا در لایه i برای پیاده سازی الگوریتم Backpropagation می باشد.

فیلد MSE هم نمایش دهنده مجموع مربعات خطا می باشد.

فیلد delta\_bias و delta\_W به ترتیب برای آپدیت کردن وزن های مربوط به بایاس و وزن های هر لایه بکار می روند.

فیلدهای 'big\_delta', DW , 'big\_delta\_bias' نگهدارنده وزن ها برای استفاده در روش آموزش دسته ای مورد استفاده قرار می گیرد.

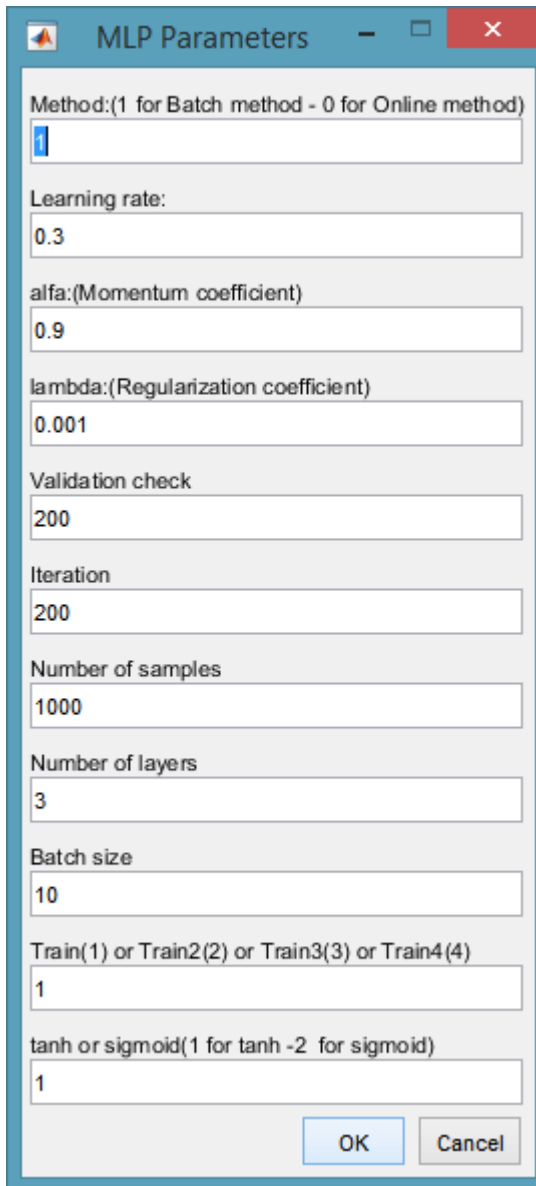
فیلد 'delta\_W\_last' برای اضافه کردن Momentum به کار می رود.

در شکل ۴ ساختار مربوط به یک MLP را با ۴ لایه می بینید:

Fields	Size	wts	z	a	delta	DW	bias	MSE	delta_W	delta_bias	big_delta	big_delta_bias	delta_W_last
1	784 []	[]		784x1000 do...	[]	[]	[] []		[]	[] []	[]	[]	[]
2	25 785x25 dou...	1x25 double	1x25 double	1x25 double	785x25 dou...		1 []		[]	0 785x25 double	1x25 double	785x25 double	
3	10 26x10 double	[-5.9156,-5....	[0.0027,0.00...	[0.0027,0.00...	26x10 double		1 10x1000 da...		[]	0 26x10 double	[-2.8662e+03,-1.7...	26x10 double	
4			activation's										

Figure 4 - MLP simple Architecture details

ورودی های درخواستی که باید توسط کاربر مقدار دهی گردند، شکل ۵:



Figures - MLP Parameters

فیلد اول روش یادگیری را مشخص میکند (یادگیری دسته ای یا آنلاین)

فیلد دوم نرخ یادگیری را مشخص میکند

فیلد سوم ضریب مومنتوم را مقدار دهی می کند

فیلد چهارم ضریب متنظم سازی را مقدار دهی می کند

فیلد پنجم مشخص کننده تعداد دفعاتی است که مقدار خطای داده های ارزیابی میتوانند در مسیر حرکت به سوی افزایش دقت نسبت به خطا روی داده های آموزشی بیشتر گردد

فیلد ششم تعداد تکرار های الگوریتم یادگیری را مشخص می کند

فیلد هفتم تعداد نمونه های آموزشی

فیلد هشتم تعداد لایه ها

فیلد نهم مشخص کننده اندازه پنجره در روش mini batch می باشد

فیلد یازدهم تعیین کننده یکی از اسکریپت های یادگیری می باشد

و فیلد دوازدهم مشخص کننده تابع فعال ساز مورد استفاده می باشد

## شیوه انجام کار

- Load MNIST Dataset
- Receive parameter from user in input before start learning
- Create MLP Layer's
- Train the network

ایجاد ساختار شبکه عصبی و مقدار دهی اولیه تصادفی به وزن ها

مقدار دهی اولیه به پارامتر هایی مثل:

نرخ یادگیری، تعداد لایه ها و تعداد نرون های هر لایه، ضرب منتظم سازی، تعداد خروجی ها و ...

با توجه به ارائه حضوری فقط کلیات در اینجا ذکر شد و وارد جزئیات نخواهیم شد.

بعد از طراحی شبکه حالا نوبت به تست و استفاده از آن میرسد.

## یادگیری با روش دسته ای

نمودار دقت بر حسب تعداد تکرار:

بهترین دقت بدست آمده با ۴ لایه و ۵۰۰۰۰ داده آموزشی:

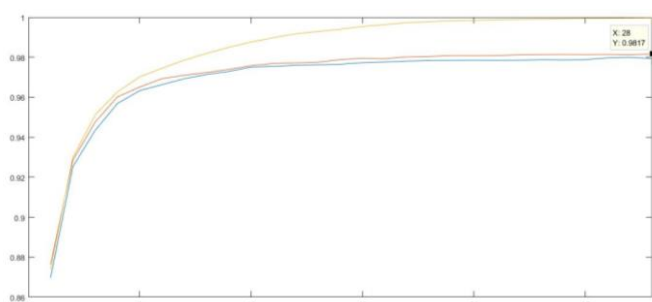


Figure 6 - Best Accuracy

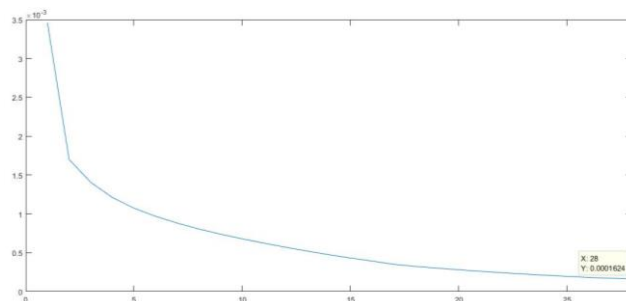


Figure 7 - Best MSE of best accuracy

حال اگر اندازه پنجره را در روش mini batch تغییر دهیم، خواهیم داشت:

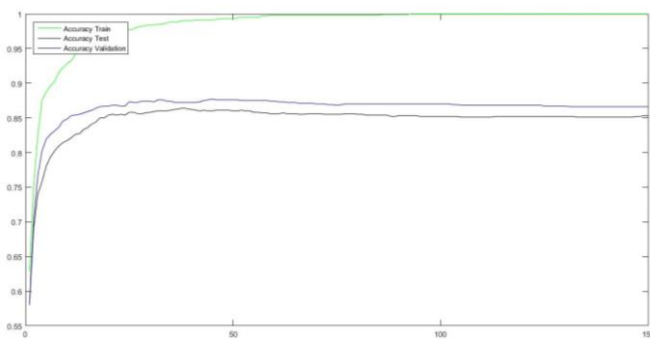


Figure 8 - Accuracy Mini batch = 10

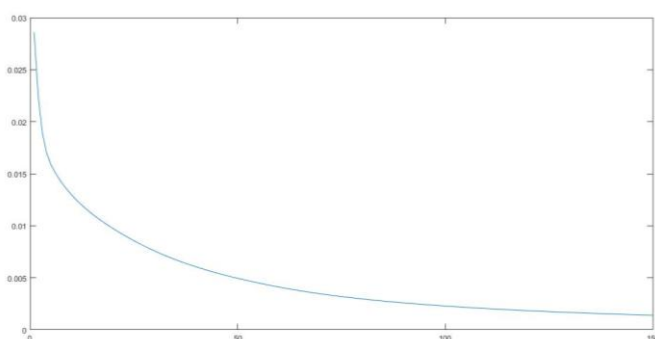


Figure 9 - MSE Mini batch = 10

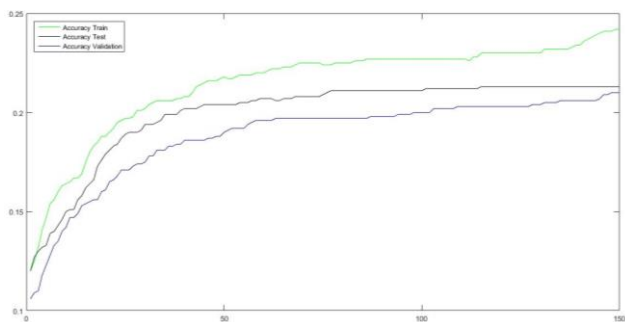


Figure 10 - Accuracy Mini batch = 1000

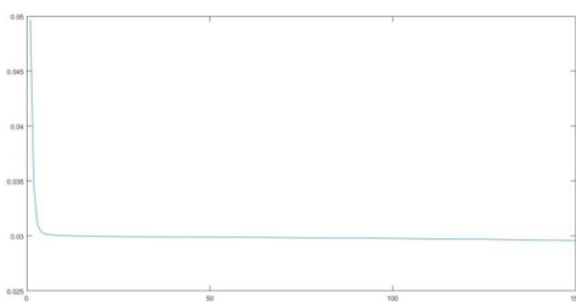


Figure 11 - MSE Mini batch = 1000

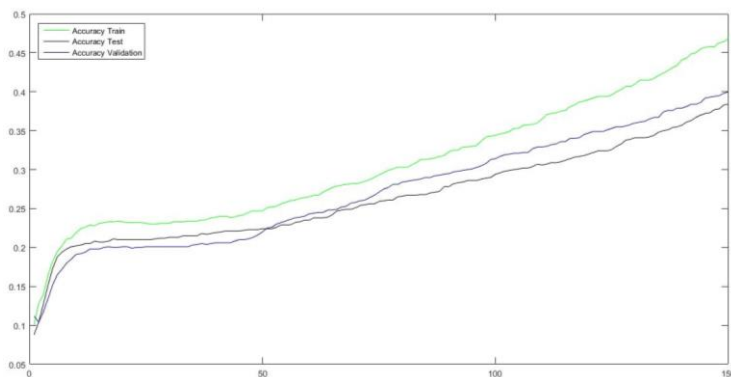


Figure 12 - Accuracy Mini batch = 500

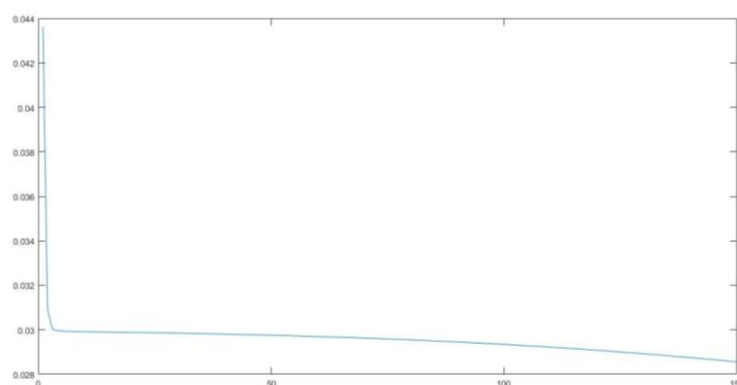


Figure 13 - MSE Mini batch = 500

Mini batch = 100

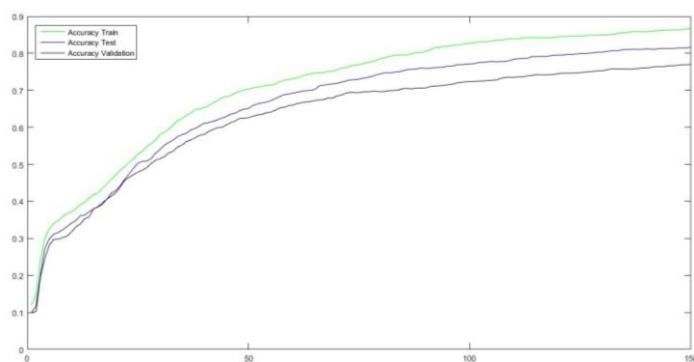


Figure 14 - Accuracy Mini batch = 100

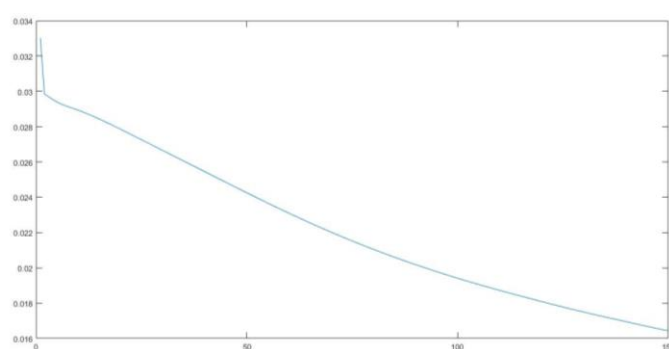


Figure 15 - MSE Mini batch = 100

مشاهده می کنید که هر چه به سمت افزایش اندازه پنجره پیش میرویم، سرعت همگرایی یا رشد دقت کاهش می یابد به صورت خلاصه جدول زیر را برای اجرای برنامه در ۱۰۰ تکرار داریم:

Table ۱ Mini batch window size

اندازه پنجره	دقت بر روی نمونه های تست	میزان خطای مجموع مربعات
۱۰	۸۵ درصد	۰,۰۰۲
۱۰۰	۷۲ درصد	۰,۰۱۷
۵۰۰	۴۰ درصد	۰,۰۲۸
۱۰۰۰	۲۰ درصد	۰,۳

با افزایش اندازه پنجره در واقع تعیین کننده تعداد داده هایی هستیم که به شبکه نشان داده میشود سپس شبکه اطلاع وزن را انجام می دهد، شبکه بعد از دیدن تعداد داده ای به اندازه پنجره، وزن ها را بروز رسانی می کند. تکه کد بروز رسانی در روش دسته ای به صورت زیر می باشد:

```

if counter == batch_size
    if parameter.method == 1 % batch method
        for i=2:L
            % Update Weight's
            layer(i).wts(1:end-1,:) = layer(i).wts(1:end-1,:) - (1/batch_size) * parameter.learning_rate .*
            layer(i).big_delta(1:end-1,:) + parameter.lambda * parameter.learning_rate * (layer(i).wts(1:end-1,:)) - ...
            (parameter.alfa * layer(i).delta_W_last(1:end-1,:));
            % Last DELTA W Weights
            layer(i).delta_W_last(1:end-1,:) = (1/batch_size) * layer(i).big_delta(1:end-1,:);
        end
    end
end

```

```

        layer(i).big_delta= zeros(layer(i-1).Size+1,layer(i).Size);
        % Update Bias
        layer(i).wts(end,:) =layer(i).wts(end,:) - (1/batch_size)* parameter.learning_rate .*
layer(i).big_delta_bias -...
        (parameter.alfa * layer(i).delta_w_last(end,:));
        % Last DELTA W Weights
        layer(i).delta_w_last(end,:) = (1/batch_size)*layer(i).big_delta_bias;
        layer(i).big_delta_bias=zeros(1,layer(i).Size);
    end
end
counter = 0;
end

```

## یادگیری با روش ترتیبی (آنلاین)

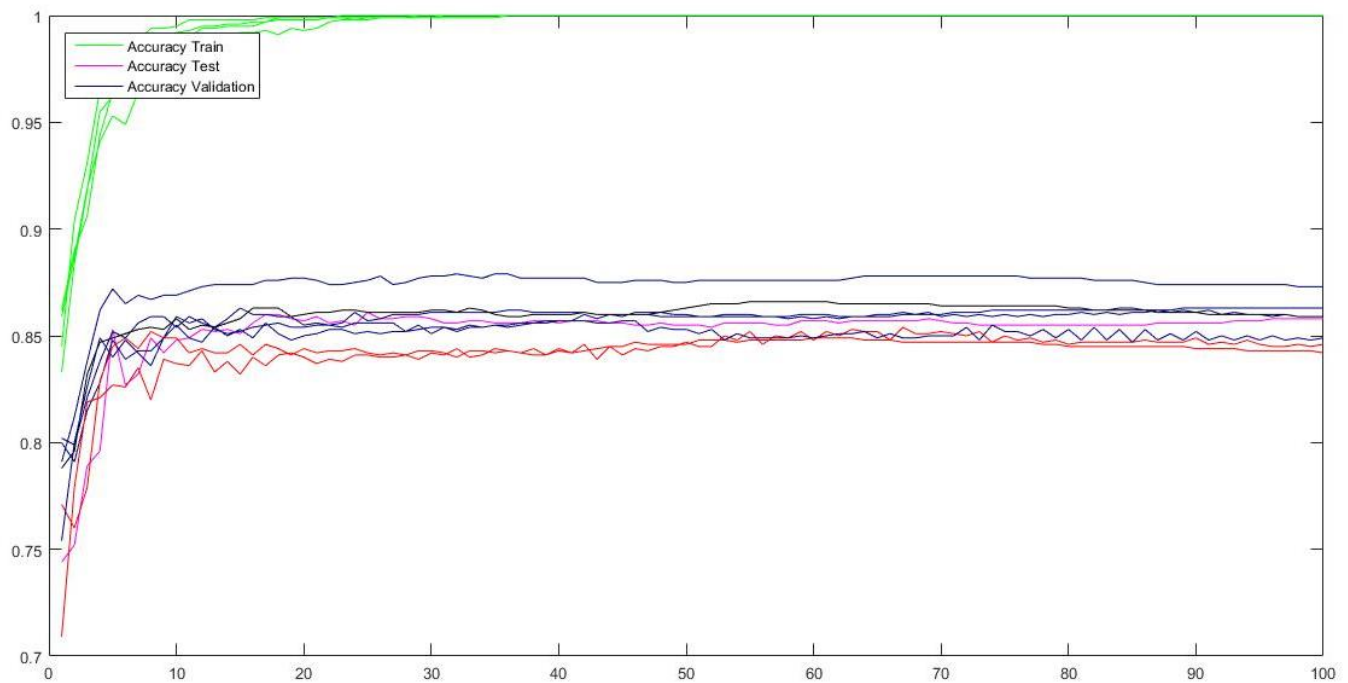


Figure 16 - Online Learning

همانطور که مشاهده میکنید در روش یادگیری ترتیبی ما با سرعت زیادی به دقت ۱۰۰ درصد بر روی داده های آموزشی میرسیم، پس سرعت همگرایی این روش نسبت به روش دسته ای بهتر است، اما به دلیل وابستگی به هر نمونه حرکتی زیگزاگی به سمت کم شدن دقت داریم.



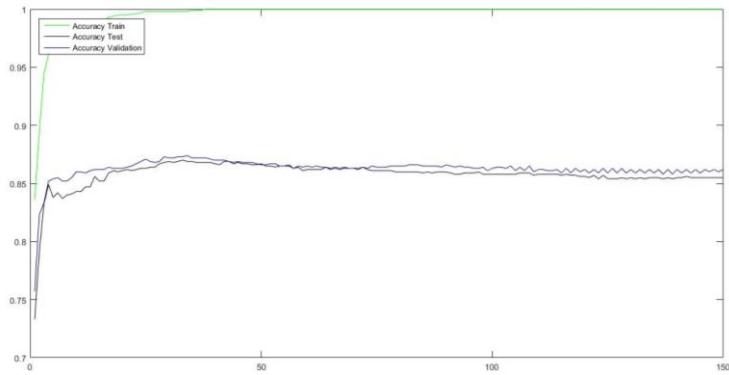


Figure 17 - Online Learning Train 3

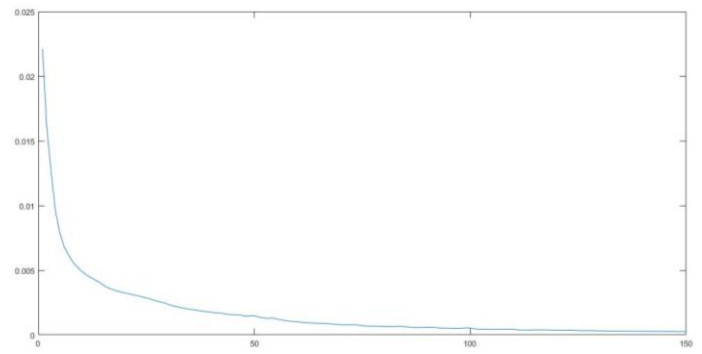


Figure 18 - MSE Online Learning

اثر منتظم سازی

$R = 1;$

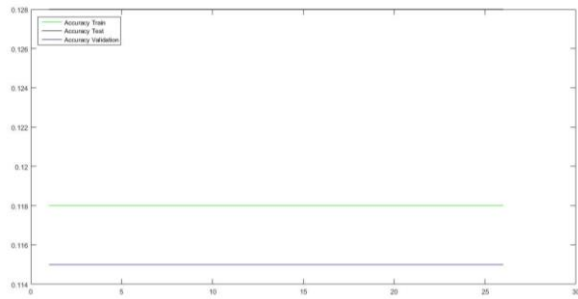


Figure 19 - Accuracy Regularization = 1

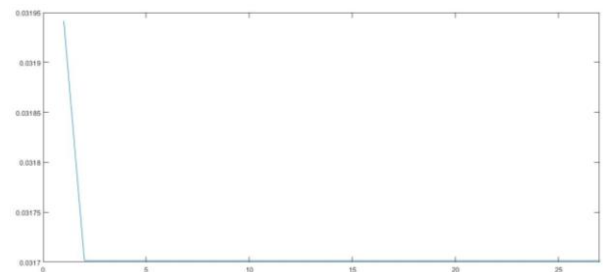


Figure 20 - MSE Regularization = 1

$R=0.5;$

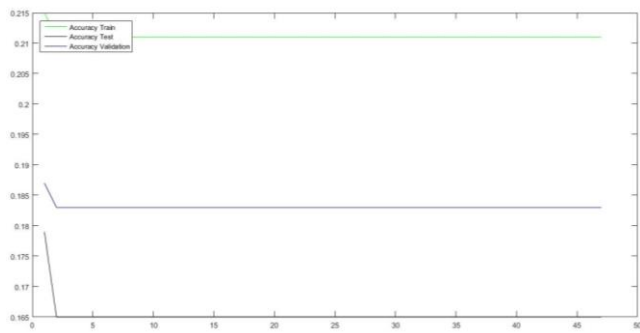


Figure 21 - Accuracy Regularization = 0.5

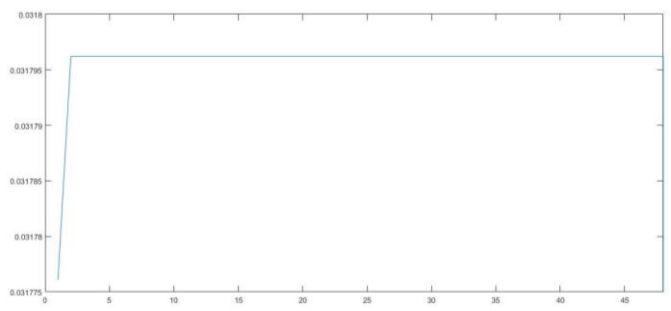


Figure 22 - MSE Regularization = 0.5

$R=0.01;$

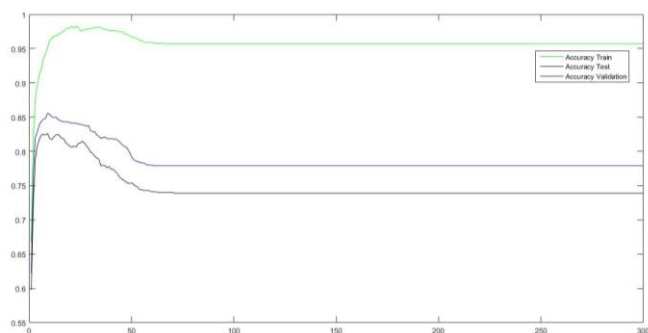


Figure 23 - Accuracy Regularization = 0.01

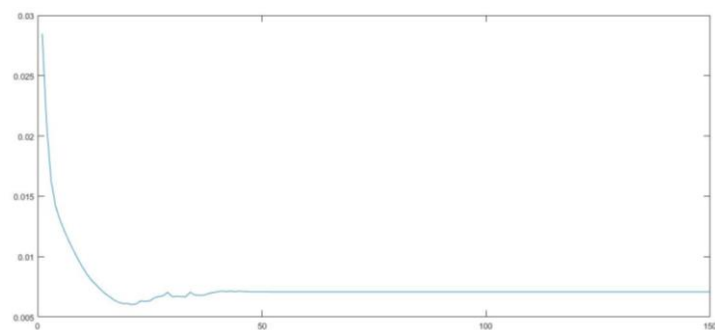


Figure 24 - MSE Regularization = 0.01

R=0.001

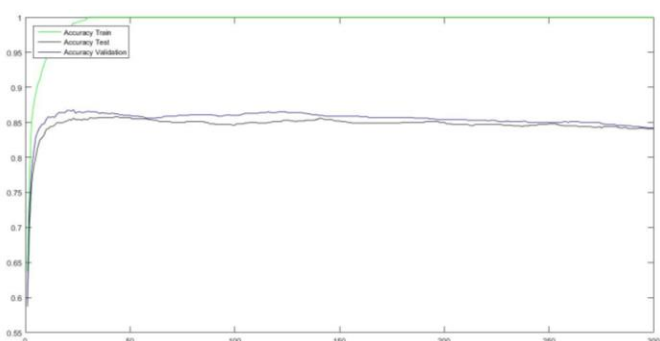


Figure 25 - Accuracy Regularization = 0.001

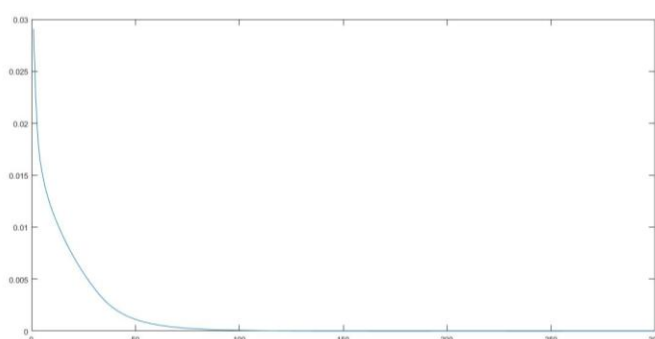


Figure 26 - MSE Regularization = 0.001

Table ۲ Regularization

میزان خطای مجموع مربعات	دقت بر روی نمونه های تست	مقدار ضریب منظم سازی
۰,۳۱۷	۱۲,۸ درصد	<b>R = 1</b>
۰,۰۳۱۷۹۶	۱۶,۵ درصد	<b>R = 0.5</b>
۰,۰۰۷	۷۴ درصد	<b>R = 0.01</b>
تقریباً صفر	۸۴ درصد	<b>R = 0.001</b>

این ضریب هم رفتاری مطابق انتظار دارد. یعنی مقدار زیاد ضریب منظم سازی باعث میشود شبکه چیزی یاد نگیرد و به سمت صفر شدن وزن ها حرکت کند، و مقدار کم این ضریب هم باعث میشود این روش تاثیر نداشته باشد و دچار بیش پوشش شویم.

### اثر تابع لجستیک

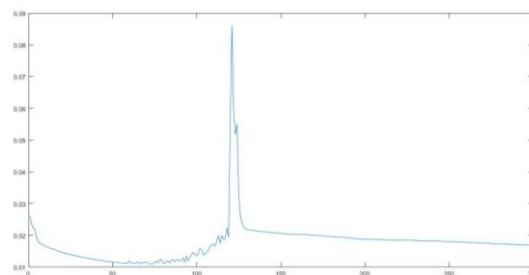
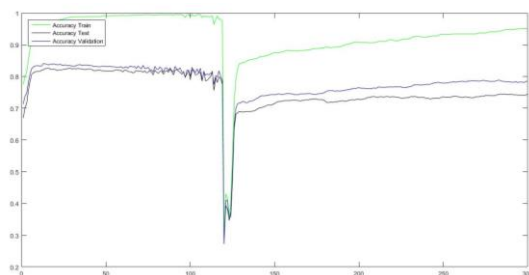


Figure 27 - Accuracy tan hyperbolic

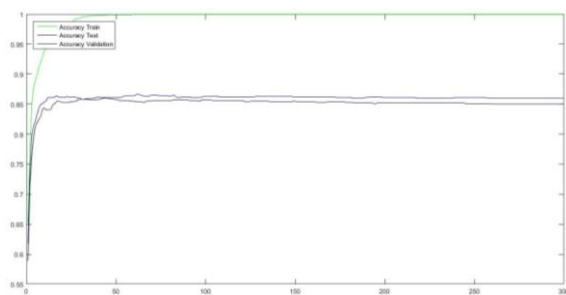


Figure 28 - MSE tan hyperbolic

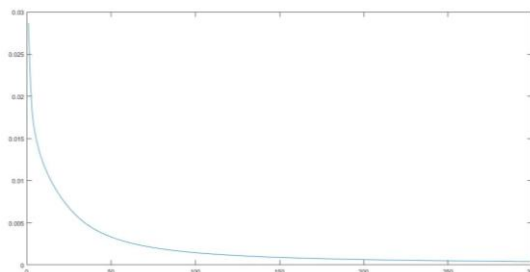


Figure 29 - Accuracy Sigmoid

Figure 30 - MSE Sigmoid

Table - Activation Function

میزان خطای مجموع مربعات	دقت بر روی نمونه های تست	تابع لجستیک
۰,۰۰۸۶۲	۷۸,۹ درصد	تانژانت هایپربولیک
۰,۰۰۳۹۶۵	۸۵ درصد	سیگموئید

در مورد اینکه کدام تابع لجستیک همواره بهتر عمل میکند نمیتوان چیزی گفت، اما میتوان گفت که در مسئله دسته بندی ما تابع سیگموئید عملکرد بهتری نسبت به تانژانت هایپربولیک دارد.

### تعداد لایه ها

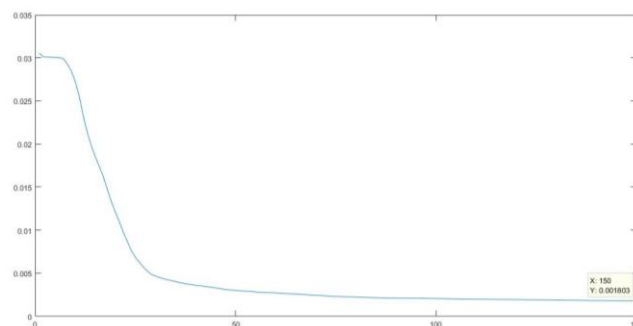
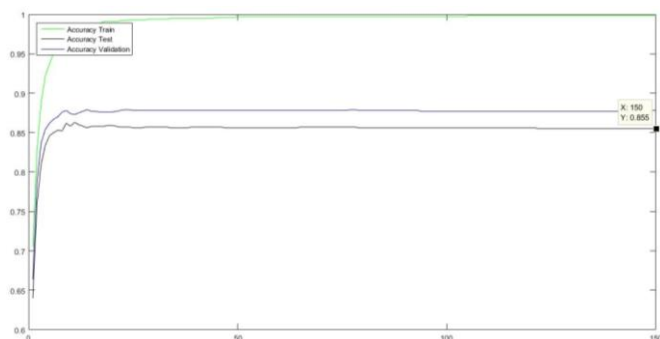


Figure 31 - Accuracy 3 Layer 784:50:10

Figure 32 - MSE 3 Layer 784:50:10

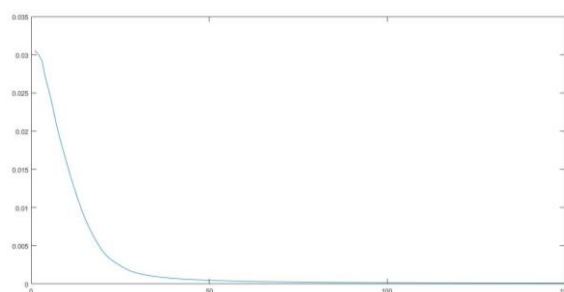
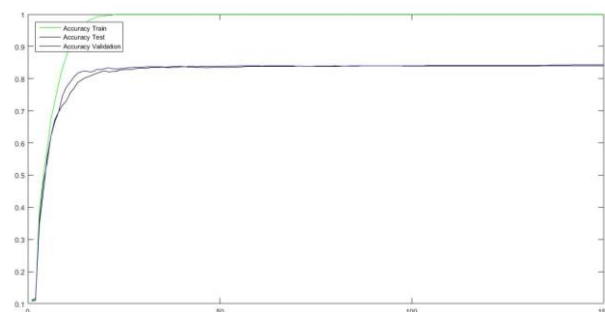


Figure 33 - Accuracy 4 Layer 784:100:50:10

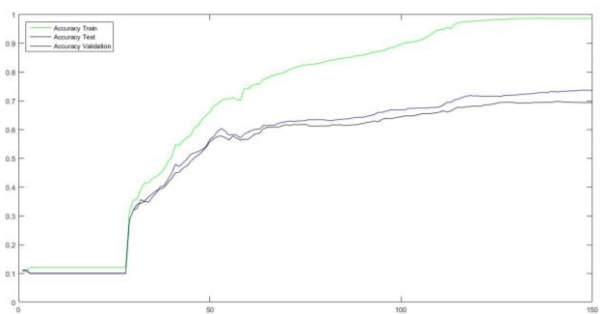


Figure 34 - MSE 4 Layer 784:100:50:10

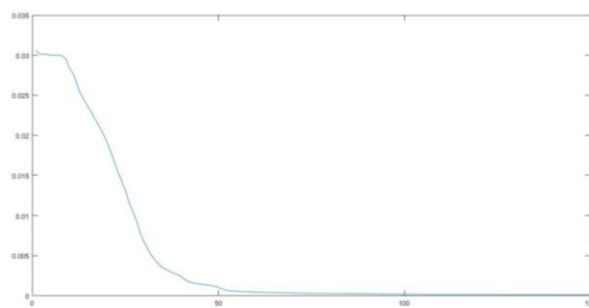


Figure 35 - Accuracy 5 Layer 784:150:100:50:10  
784:150:100:50:10

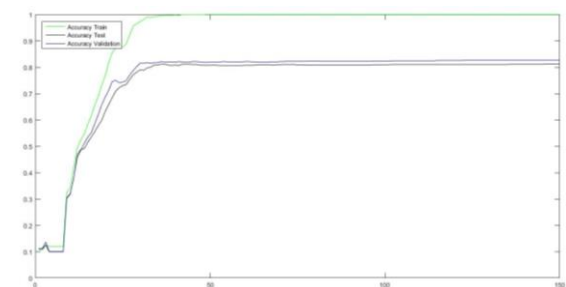


Figure 36 - MSE 5 Layer

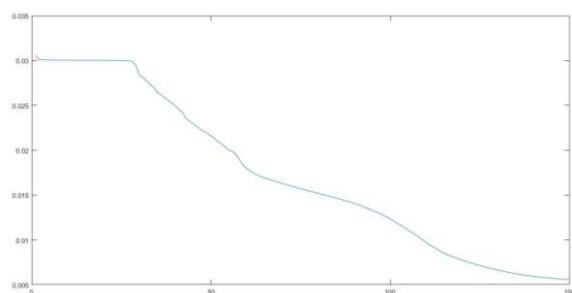


Figure 37 - Accuracy 6 Layer 784:200:150:100:50:10  
784:200:150:100:50:10

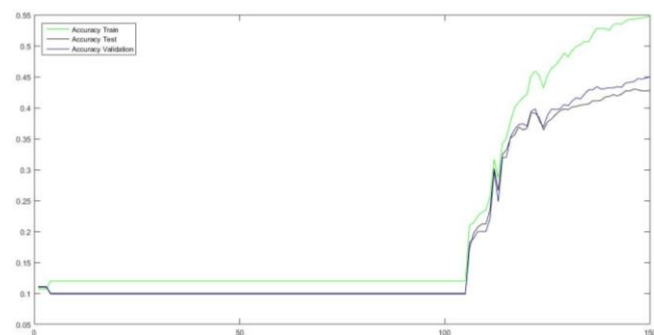


Figure 38 - MSE 6 Layer

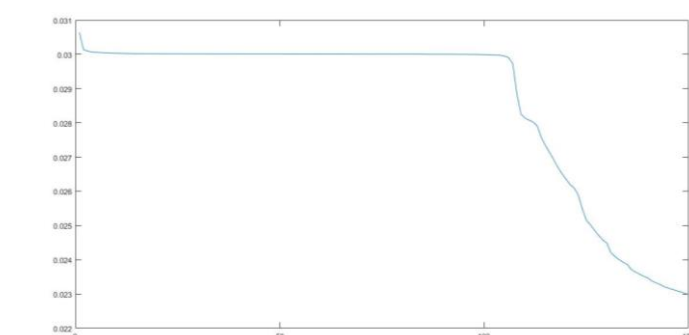


Figure 39 - Accuracy 7 Layer 784:300:200:150:100:50:10  
784:300:200:150:100:50:10

Figure 40 - MSE 7 Layer

Table 1 - Compare layer size

تعداد لایه ها	دقت بر روی نمونه های تست	میزان خطای مجموع مربعات
۷	۴۲٫۸ درصد	۰٫۰۲۲۹۸
۶	۶۹٫۳	۰٫۰۰۵۵۴۹
۵	۸۰ درصد	۰٫۰۰۱۸۰۳
۴	۸۲٫۶ درصد	۰٫۰۰۰۲۲۵۳
۳	۸۵٫۵ درصد	۰٫۰۰۴۲۹۵

با افزایش تعداد لایه ها باعث کاهش سرعت همگرایی می‌شویم، اما این کار باعث می‌شود تعداد پارامترهای آزاد افزایش یابد و امکان وقوع بیش پوشش را بیشتر می‌کند. اما بر طبق این اصل که اگر بعد داده ها نسبت به تعداد آنها بیشتر باشد باعث می‌شود که دسته بندی راحت تر صورت بگیرد، یعنی ممکن است داده ها در فضای اولیه قابل جدا سازی نباشند و اگر تعداد بعد داده ها زیاد گردد میتوان با یک دسته بند خطی این کلاس ها را از هم جدا کرد.

## تشخیص اعداد توسط CNN

برای پیاده سازی این شبکه از چارچوب MatConvNet استفاده کردیم که نتیجه آن به صورت زیر می باشد:

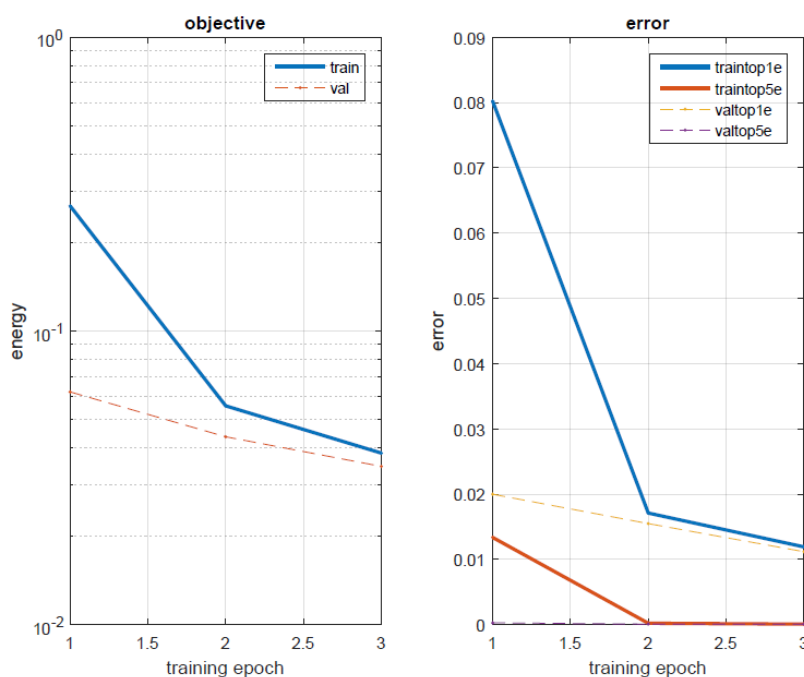


Figure 41 - Accuracy CNN

همانگونه که مشاهده میکنید با استفاده از شبکه های عصبی کانولوشن که یک نوع شبکه عمیق می باشد، ما به دقت ۹۹,۹ درصد رسیده ایم. (خط زرد رنگ)

شبکه طراحی شده برای این کار به صورت زیر است:

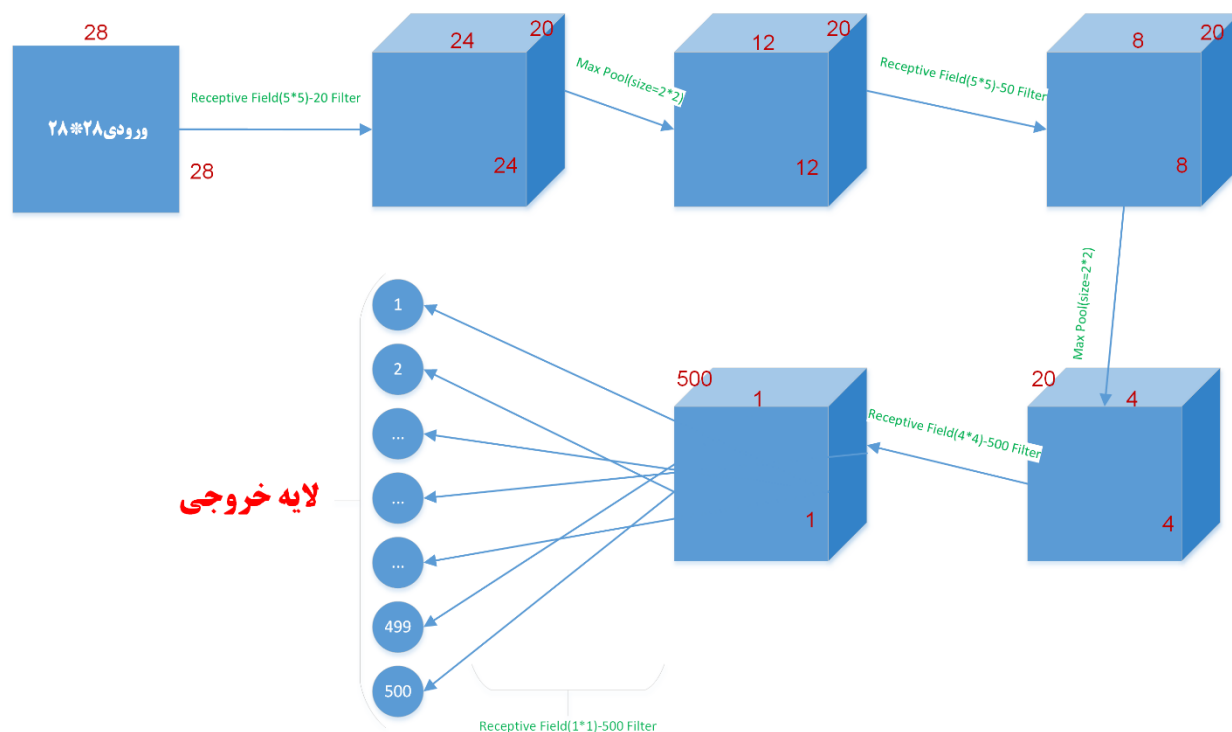


Figure 42 - CNN Architecture

همانگونه که مشاهده می کنید این شبکه قدری پیچیده شده، اما همین پیچیدگی ما را در رسیدن به نتایج بهتر موفق می کند.

## نتیجه گیری

در این پروژه ما شبکه عصبی را برای تشخیص اعداد دست نویس انگلیسی بکار بردیم، در ابتدا با یک شبکه عصبی چند لایه و سپس توسط یک شبکه عمیق همین کار را انجام دادیم و دیدیم که دقت بدست آمده توسط شبکه عصبی چند لایه (MLP [1]) برابر با ۹۸,۱۷ درصد بر روی داده های تست است و دقت شبکه عمیق برابر با ۹۹,۹ درصد بر روی داده های تست است.

## مراجع

- [1] "ufldl.stanford.edu," [Online]. Available: [http://ufldl.stanford.edu/wiki/index.php/UFLDL\\_Tutorial](http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial).
- [2] "MatConvNet," [Online]. Available: [www.vlfeat.org/matconvnet/](http://www.vlfeat.org/matconvnet/).
- [3] "neural networks and deep learning," [Online]. Available: <http://neuralnetworksanddeeplearning.com/>.