e

**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 جند لایه با قدمتی بیش از 60 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 10 سال میرسد استفاده نماییم(منظور از 10 سال این نیست که قبلا شبکه های عمیق وجود نداشته اند، بلکه قبلا به دلیل مشکلاتی که اینگونه شبکه ها داشتند، کنار گذاشته شده بودند و مجددا 10 سال است که مورد استفاده قرار گرفته اند وکارایی 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 CNN [3]) 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 نورون های هر لایه، تعداد ورودی ها و خروجی ها و سایر پارامتر ها 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 نورون 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).

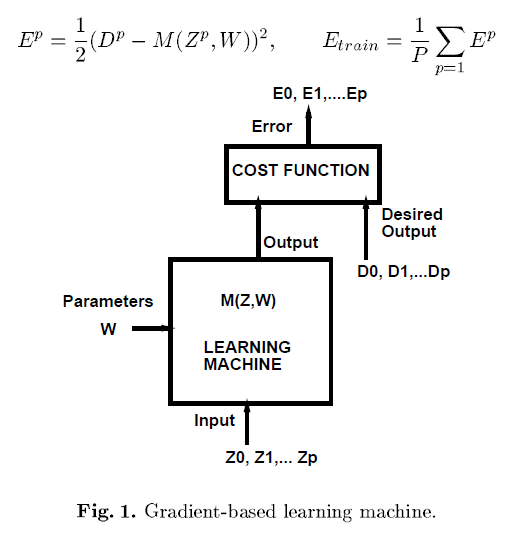


Figure - Gradient-based Learning Machine

تئوری یادگیری گرادیانی با ناظر ما در شکل 1 خلاصه شده است.

بعد از بدست آمدن مقدار خطا با روش Backpropagation [3] 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 4 لایه ای که دارای 784 ورودی و 10 خروجی و 2 لایه مخفی می باشد مانند شکل 2 می باشد.

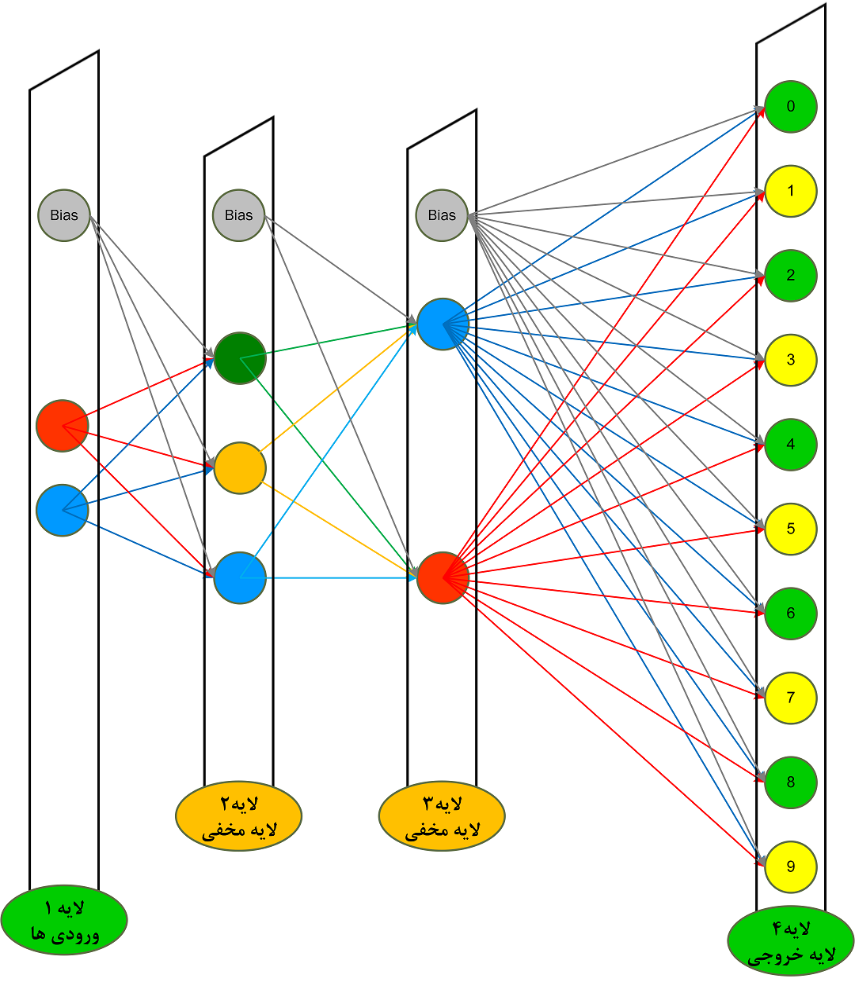


Figure - MLP simple Architecture

مدل ما 4 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 10 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.

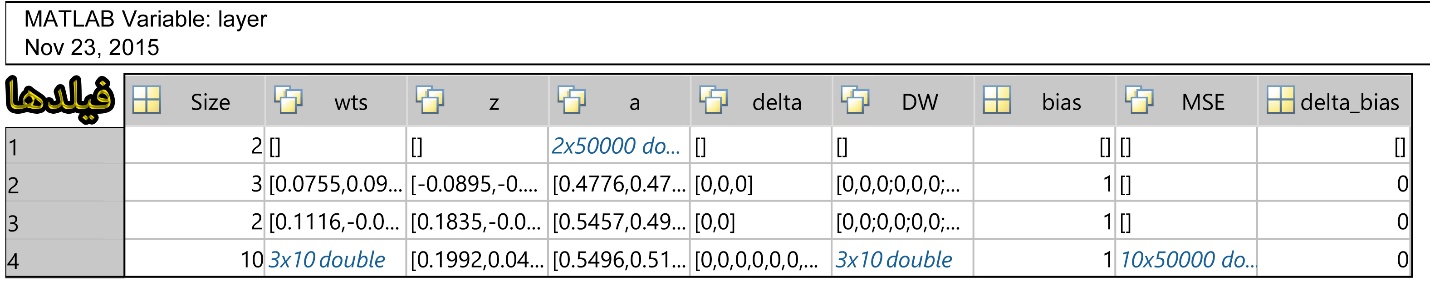
شکل 3 نشان دهنده ساختار ساده شده شبکه طراحی شده است:

Figure - 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 نورون ها در لایه iام 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 دوبعدی 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 نورون j در لایه i می باشد.(z=WX)

فیلد layer(i).a مشخص کننده اعمال تابع سیگموید بر روی مقدار layer(i).z می باشد.(a=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\_bias', DW , 'big\_delta'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.

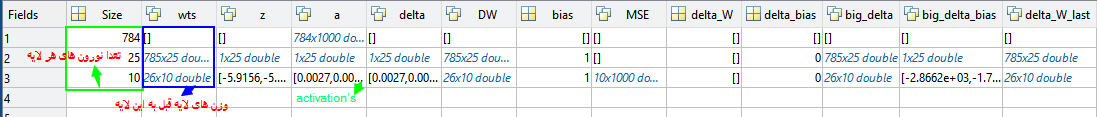
در شکل 4 ساختار مربوط به یکMLP را با 4لایه می بینید:

Figure - MLP simple Architecture details

ورودی های درخواستی که باید توسط کاربرمقدار دهی گردند،شکل 5:

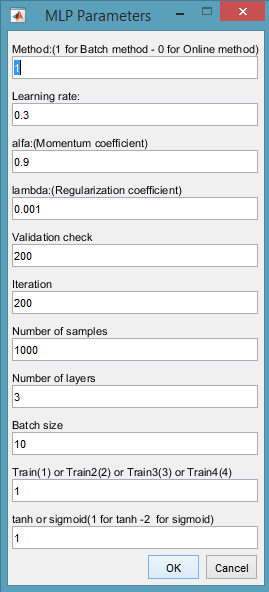


Figure 5 - MLP Parameters

فیلد اول روش یادگیر ی 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 متنطم 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:

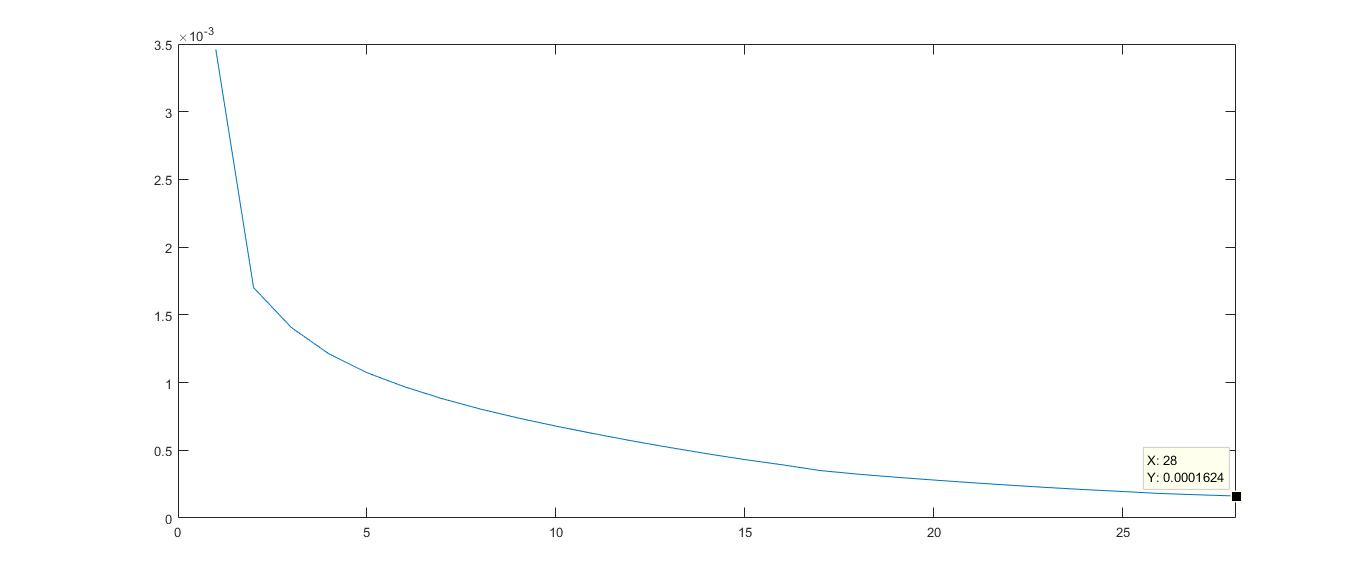
نرخ یادگیری، تعدا لایه ها و تعدا نورون های هر لایه، ضریب منتظم سازی، تعداد خروجی ها و ...

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 4 لایه و 50000 داده آموزشی:

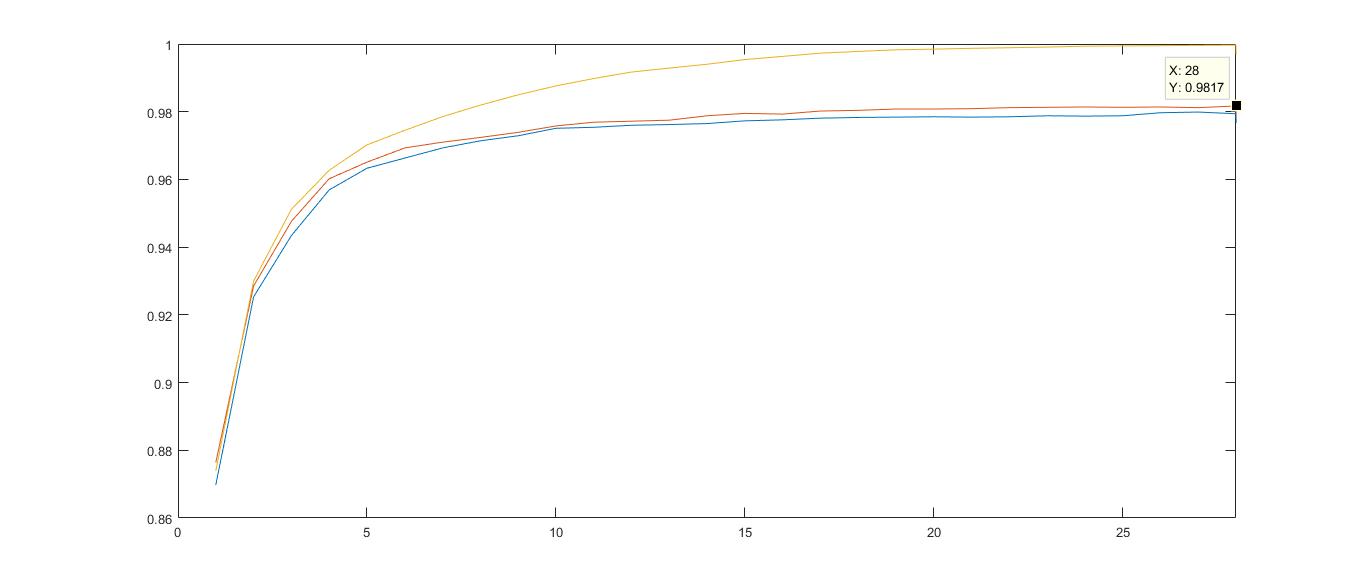


Figure - Best Accuracy Figure - Best MSE of best accuracy

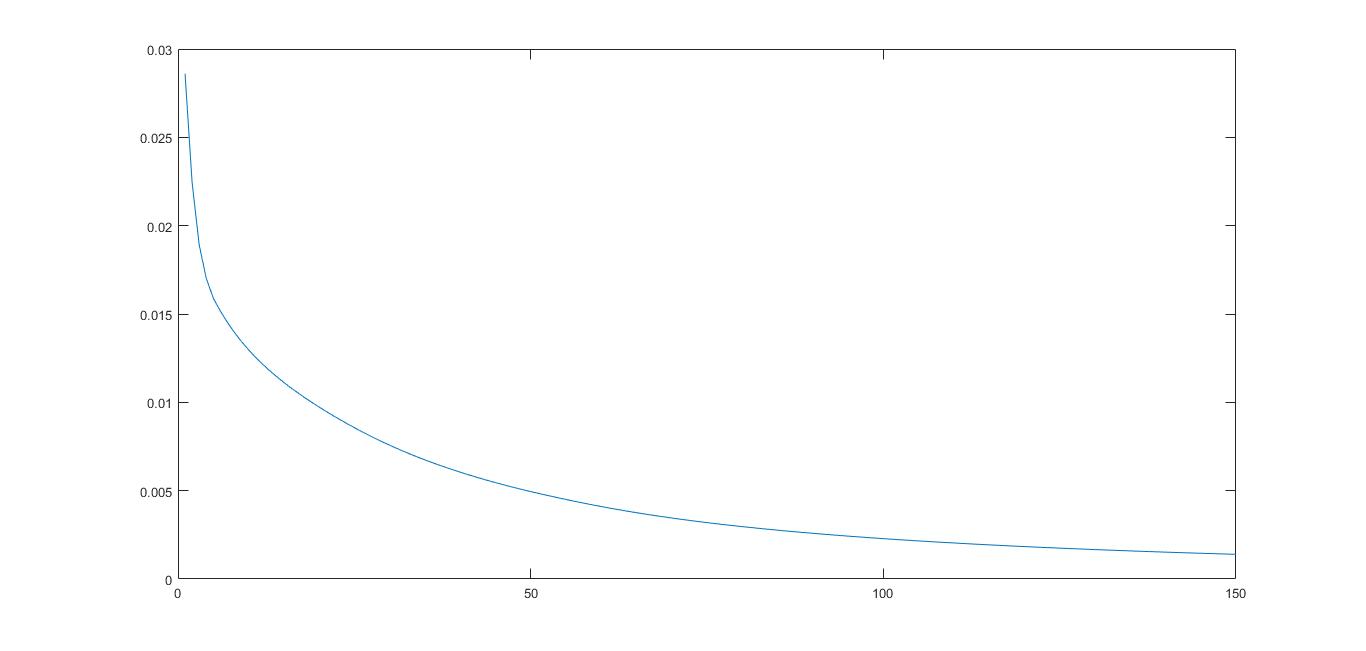
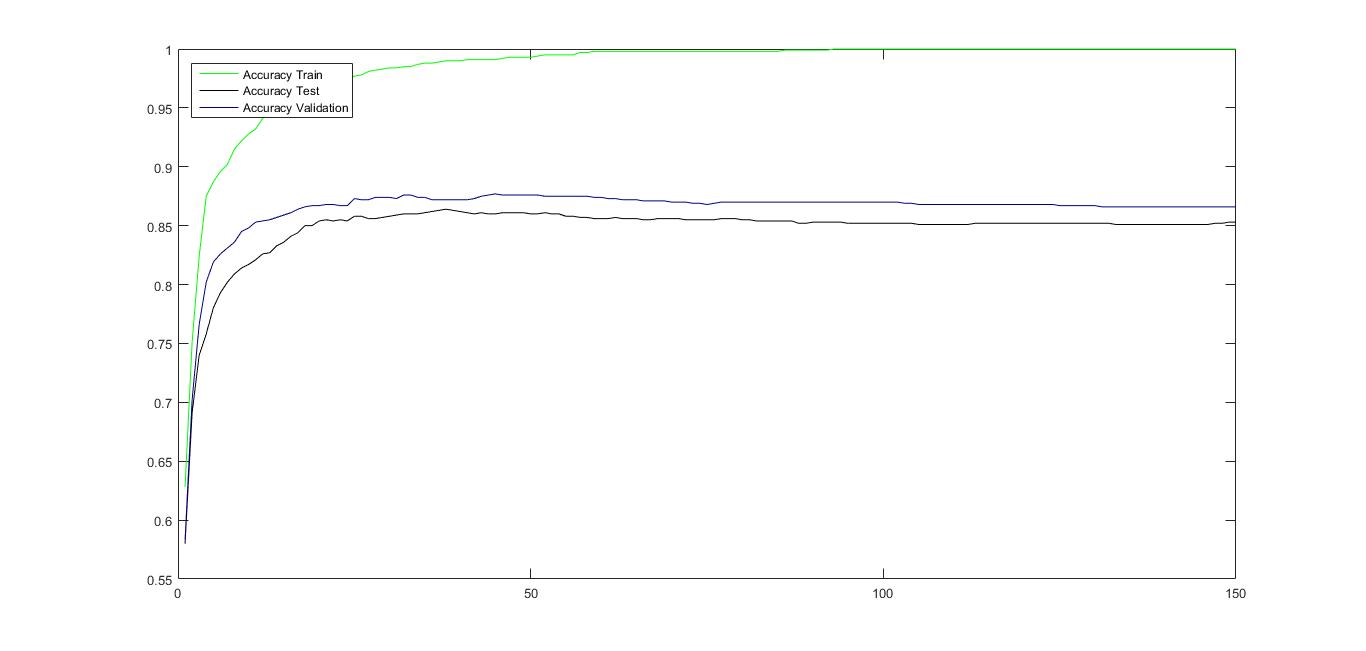
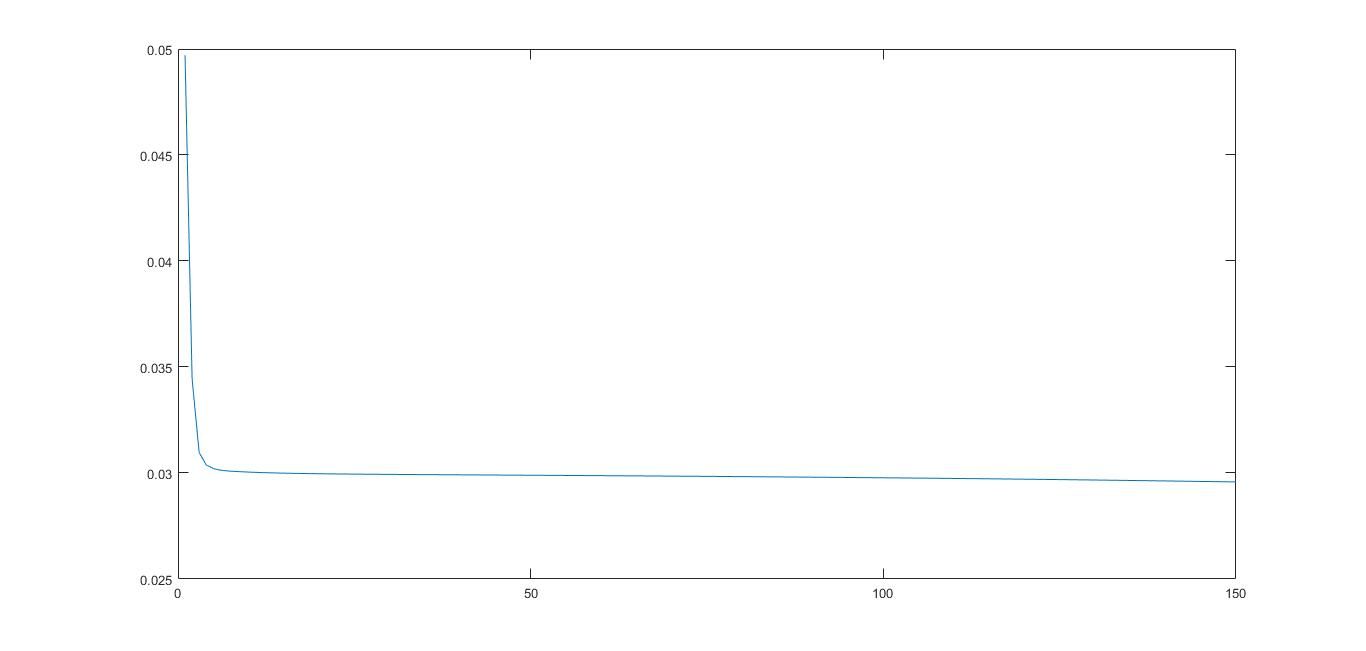
حال اگر اندازه پنجره را در روش mini batch تغییر دهیم، خواهیم داشت:

Figure – Accuracy Mini batch = 10 Figure - MSE Mini batch = 10



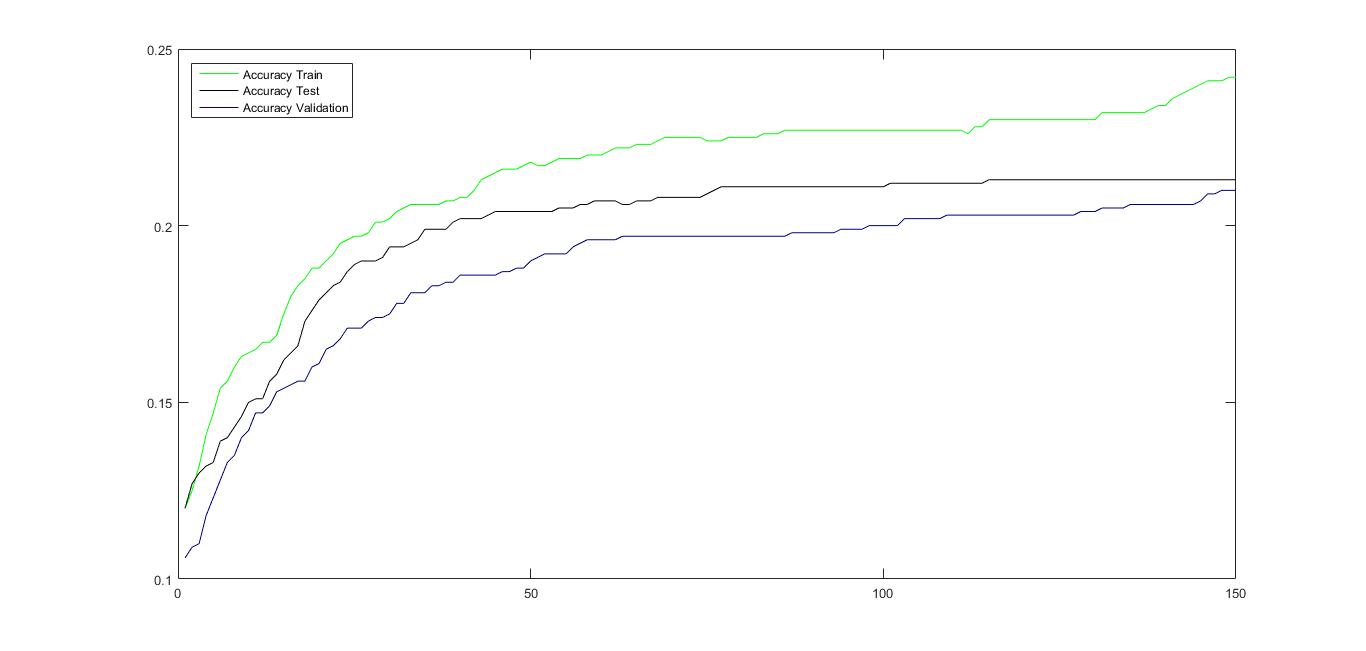


Figure - Accuracy Mini batch =1000 Figure - MSE Mini batch =1000

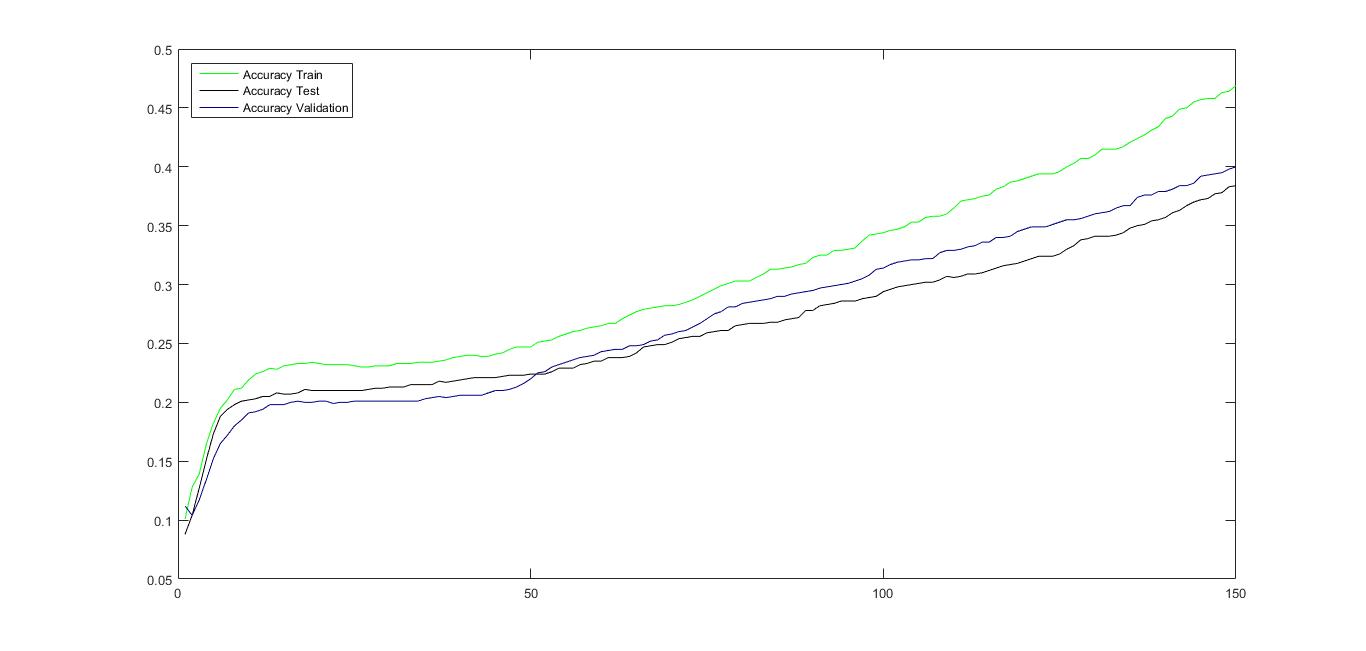
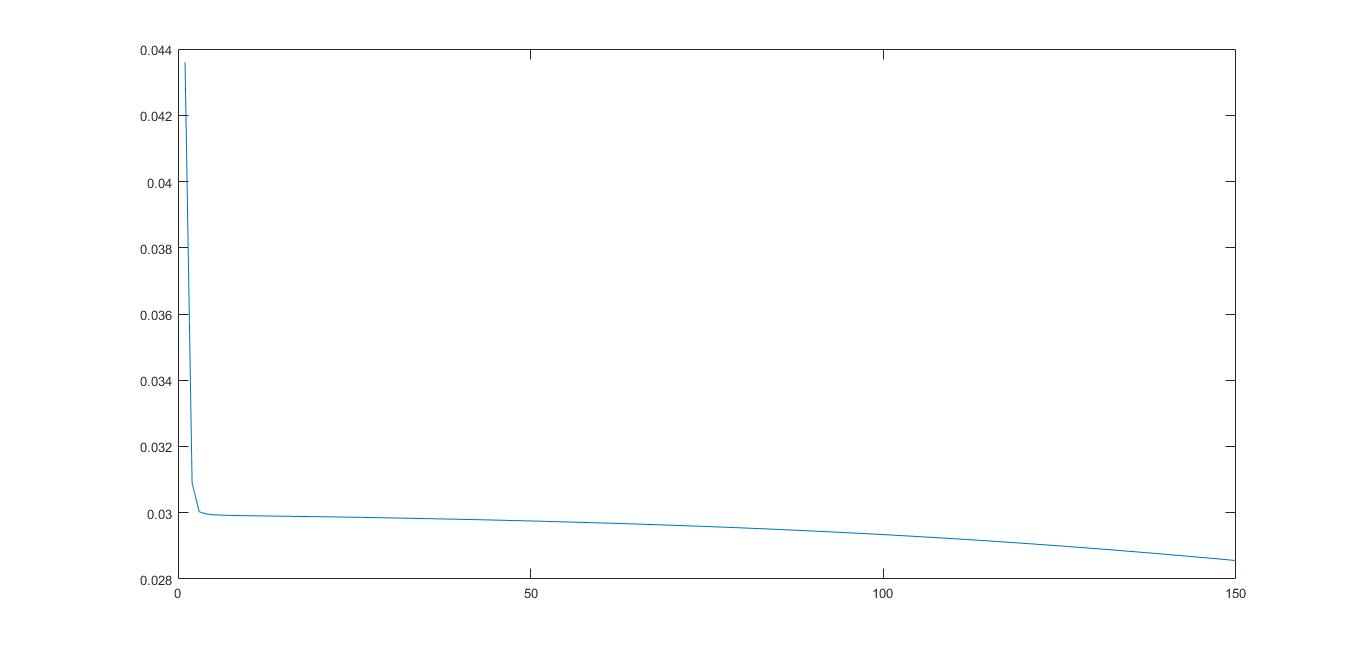


Figure - Accuracy Mini batch = 500 Figure - MSE Mini batch = 500

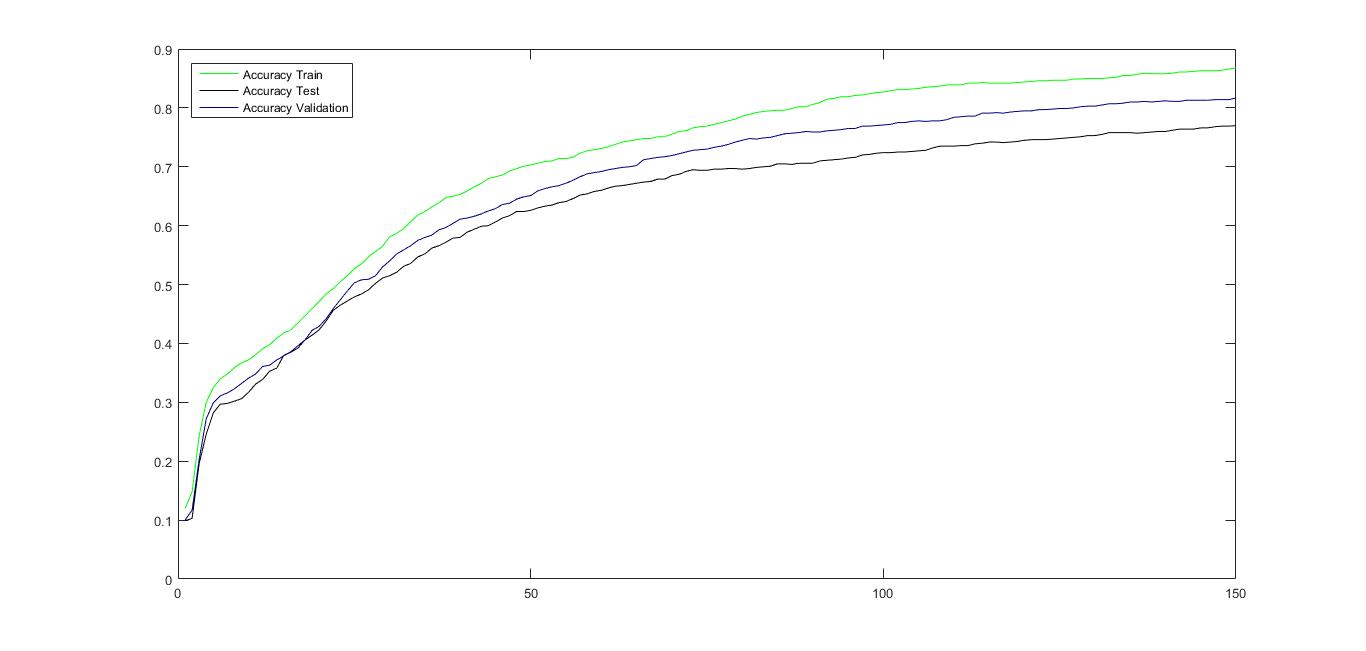
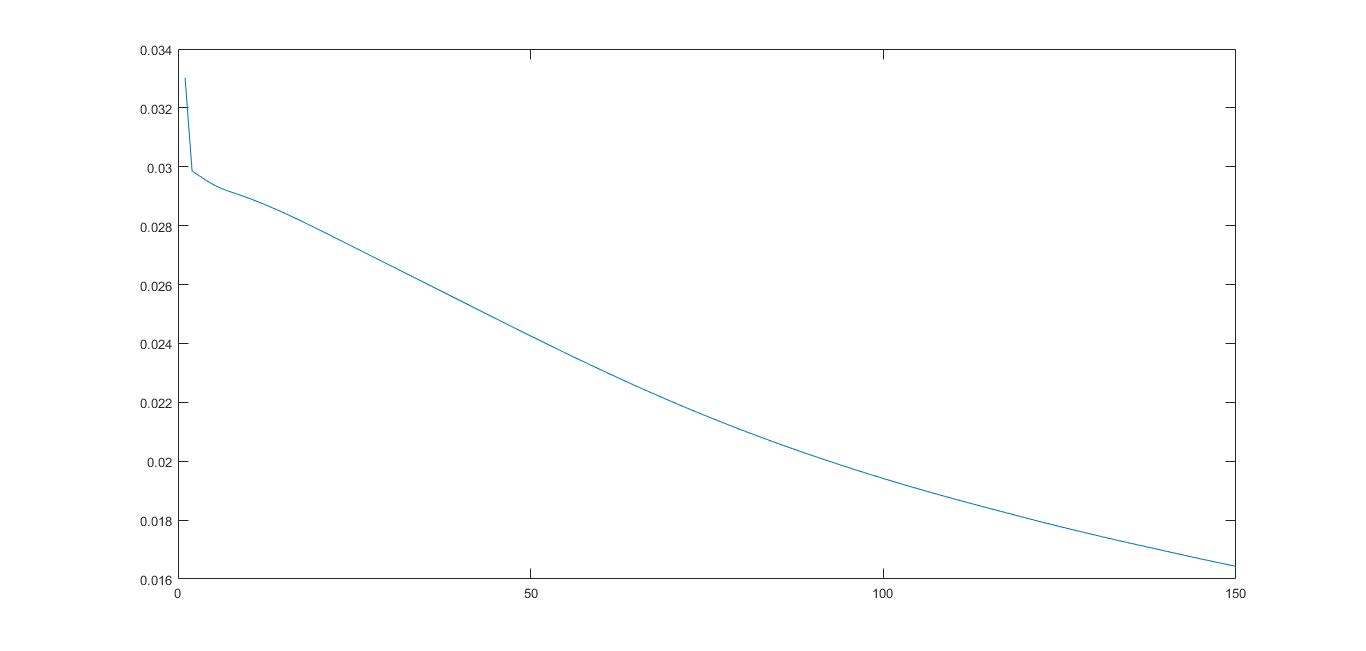
Mini batch = 100

Figure - Accuracy Mini batch = 100 Figure - 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 100 تکرار داریم:

Table 1 Mini batch window size

|  |  |  |
| --- | --- | --- |
| اندازه پنجره | دقت بر روی نمونه های تست | میزان خطای مجموع مربعات |
| 10 | 85 درصد | 0.002 |
| 100 | 72 درصد | 0.017 |
| 500 | 40 درصد | 0.028 |
| 1000 | 20 درصد | 0.3 |

با افزایش اندازه پنجره در واقع تعیین کننده تعداد داده هایی هستیم که به شبکه نشان داده میشود سپس شبکه اطلاح 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,:);

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

# F:\Documents\MATLAB\Machine Learning\MR ML Project\4-2.jpgیادگیری با روش ترتیبی(آنلاین)

Figure 16 - Online Learning

: همانطور که مشاهده میکنید در روش یادگیری ترتیبی ما با سرعت زیادی به دقت 100درصد 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.

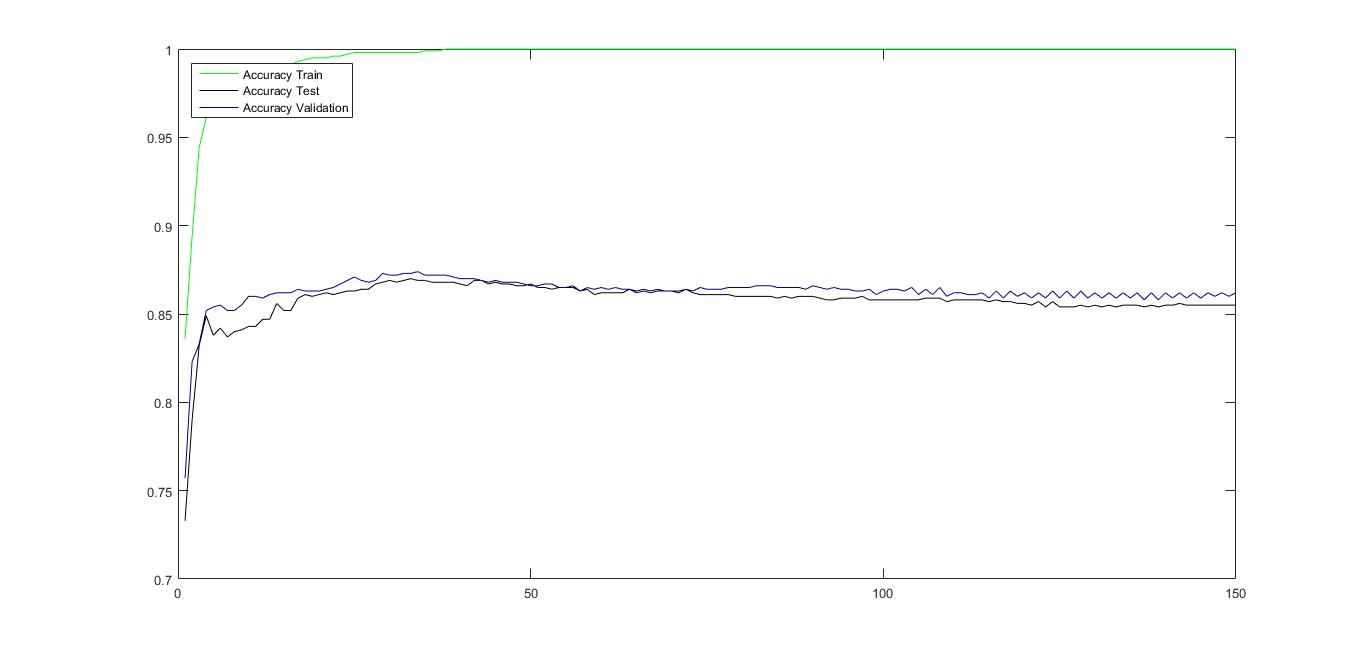
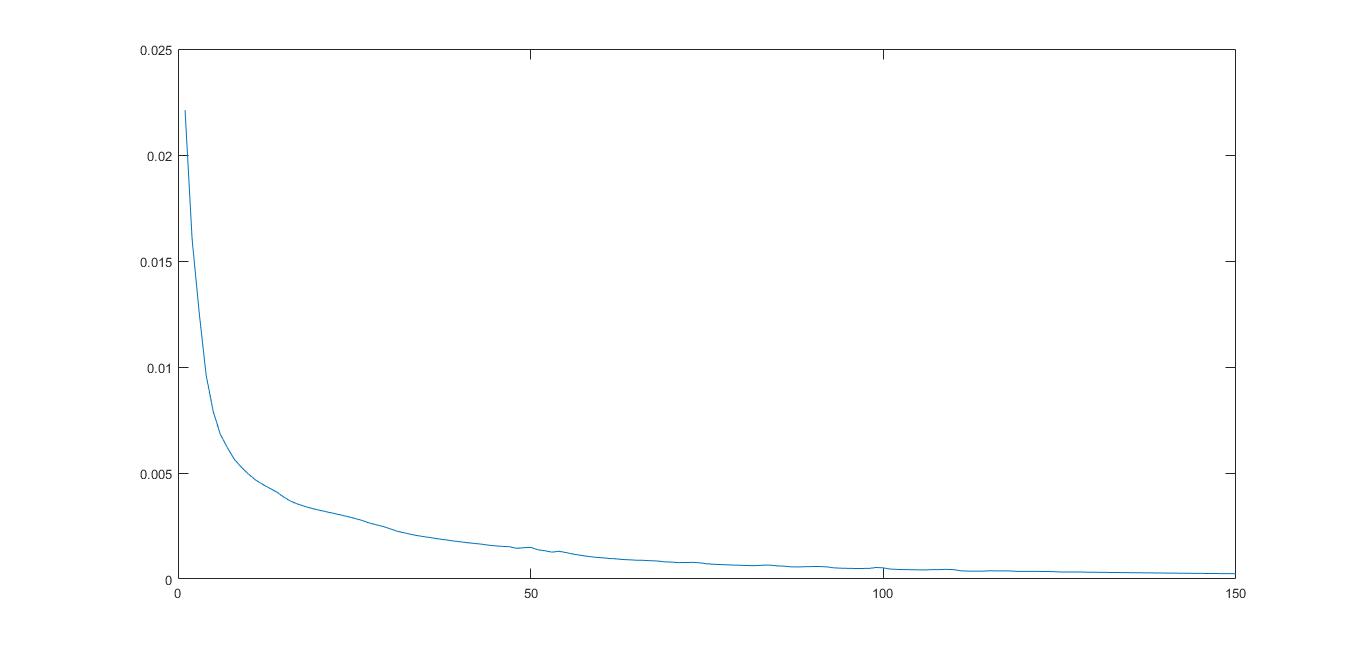


Figure - Online Learning Train 3 Figure - MSE Online Learning

# اثر منتظم سازی

R = 1;

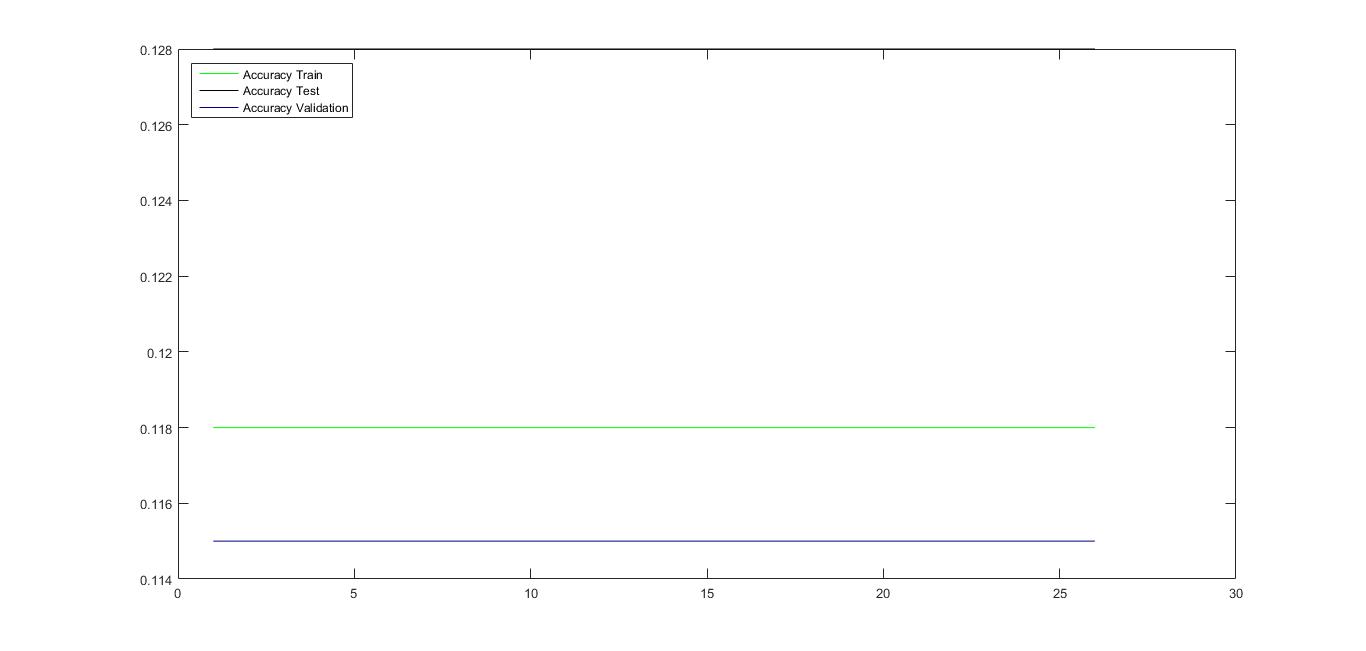
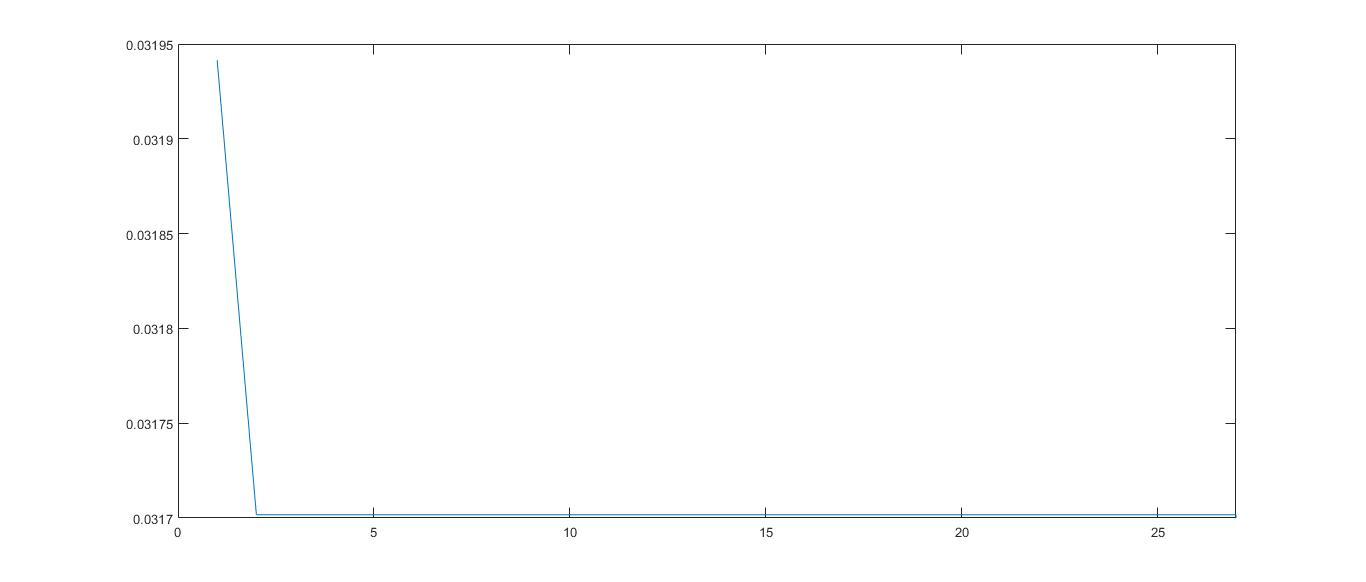
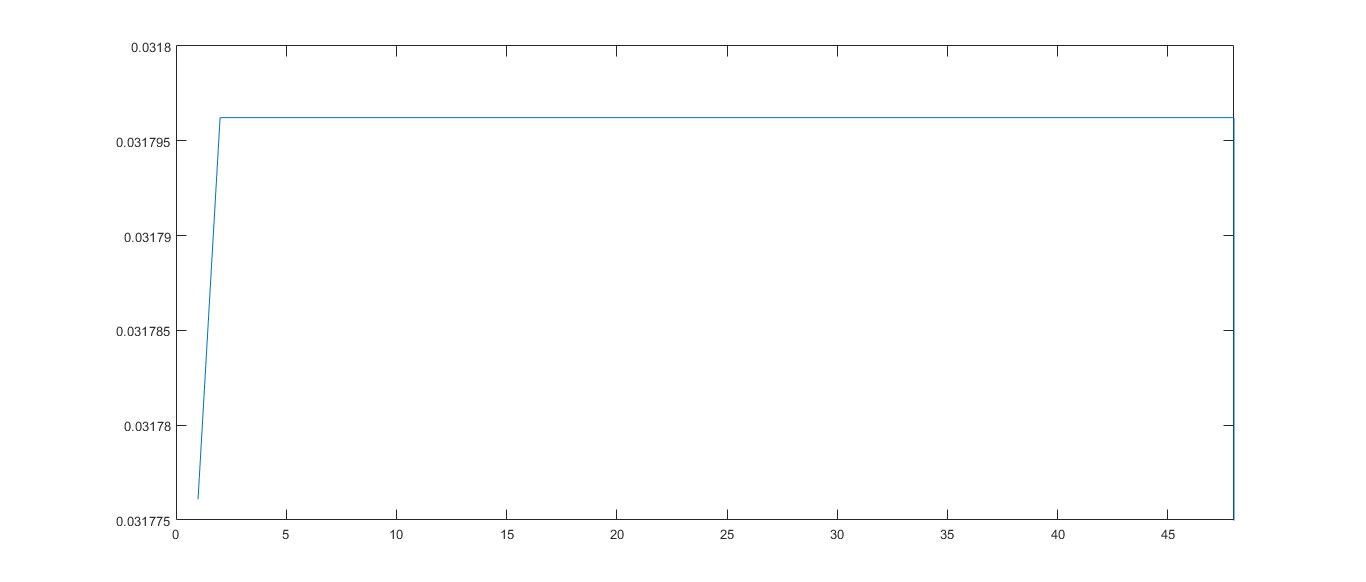


Figure - Accuracy Regularization = 1 Figure - MSE Regularization = 1

R=0.5;

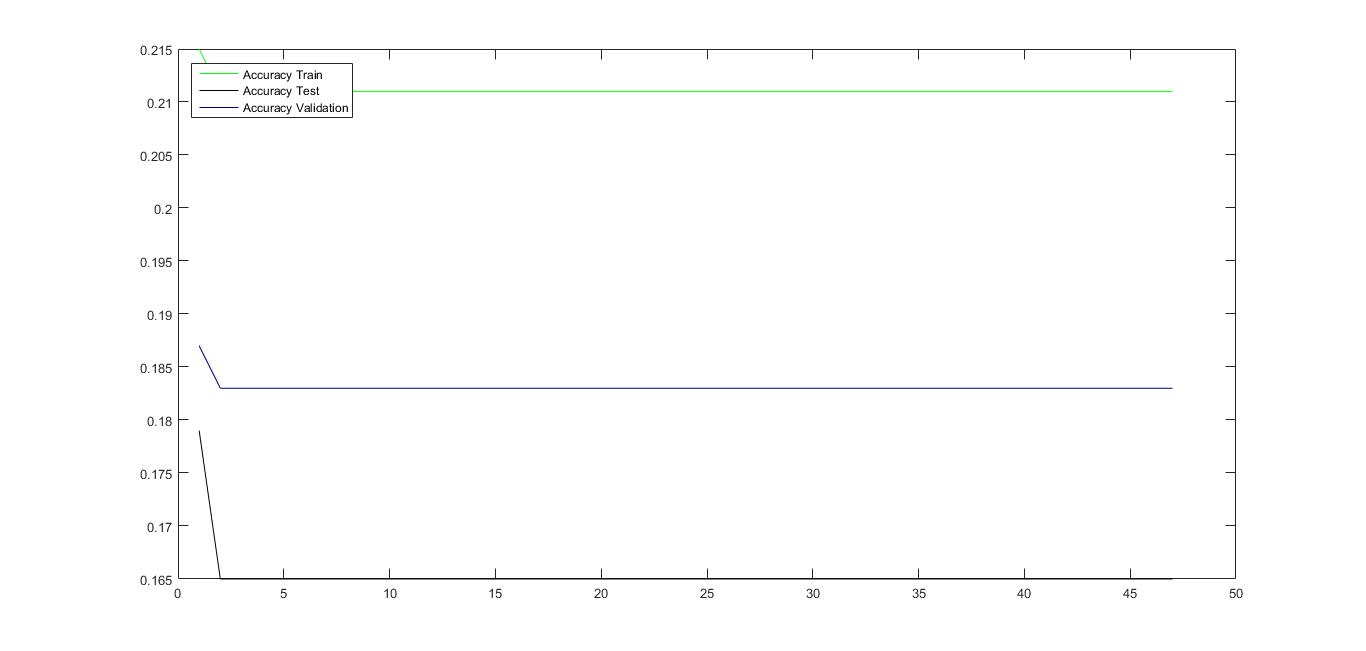


Figure - Accuracy Regularization = 0.5 Figure - MSE Regularization = 0.5

R=0.01;

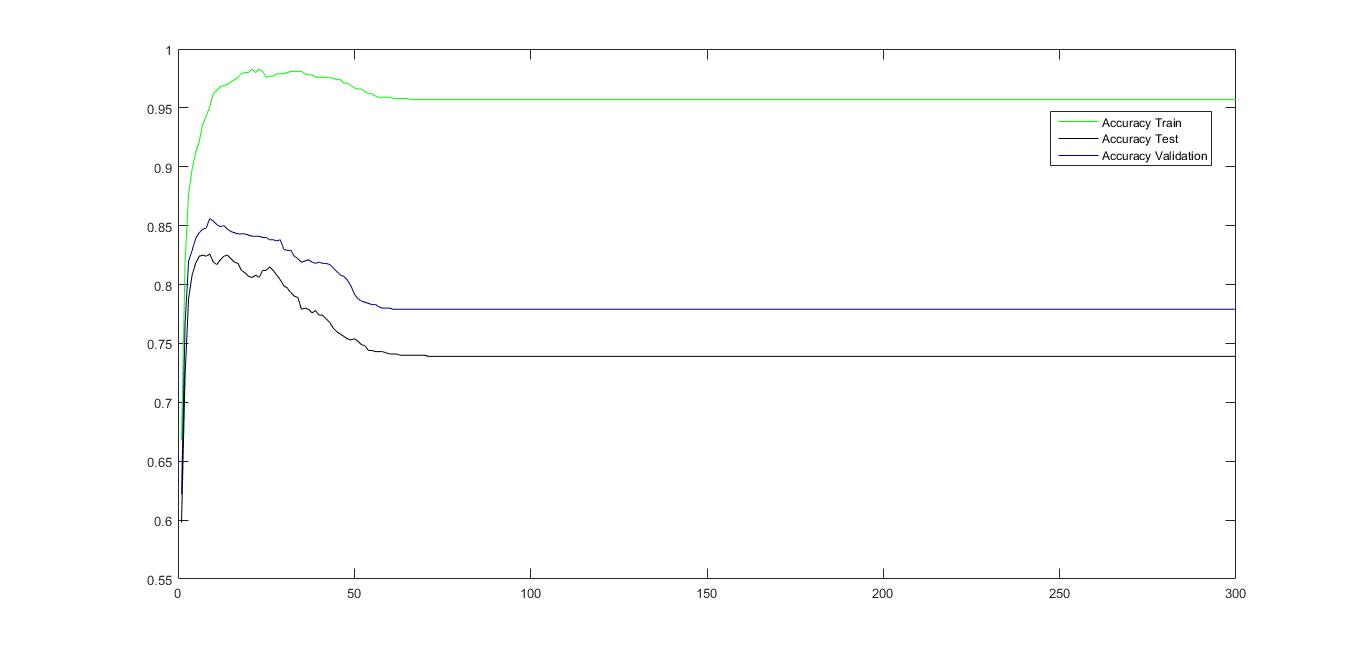
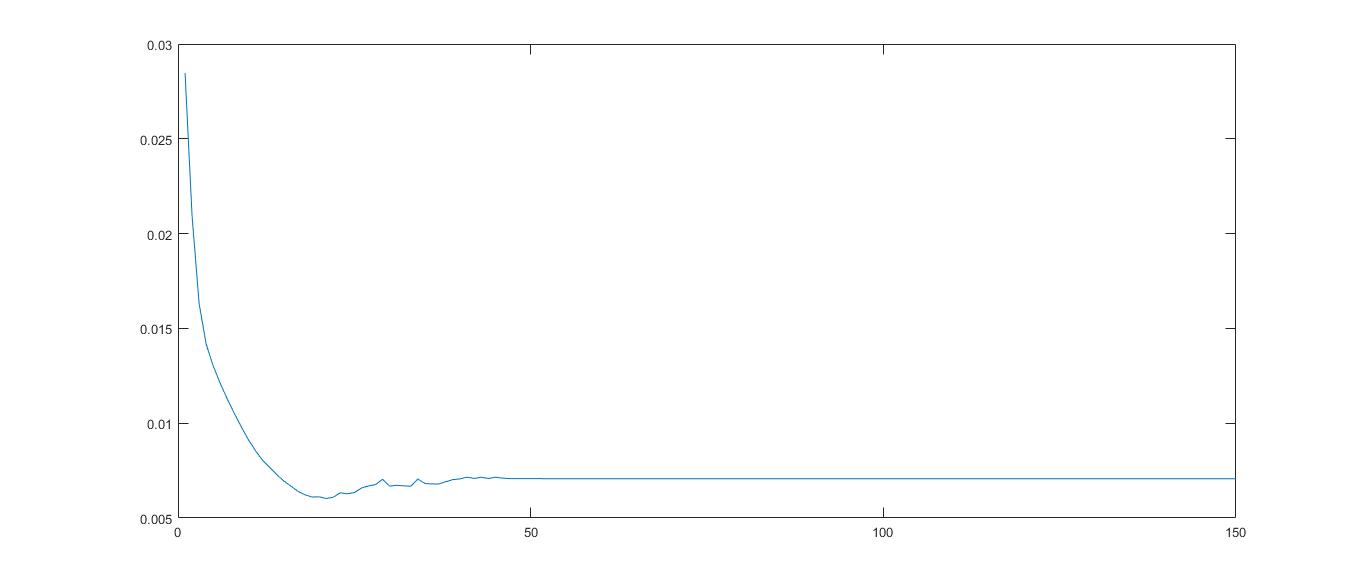
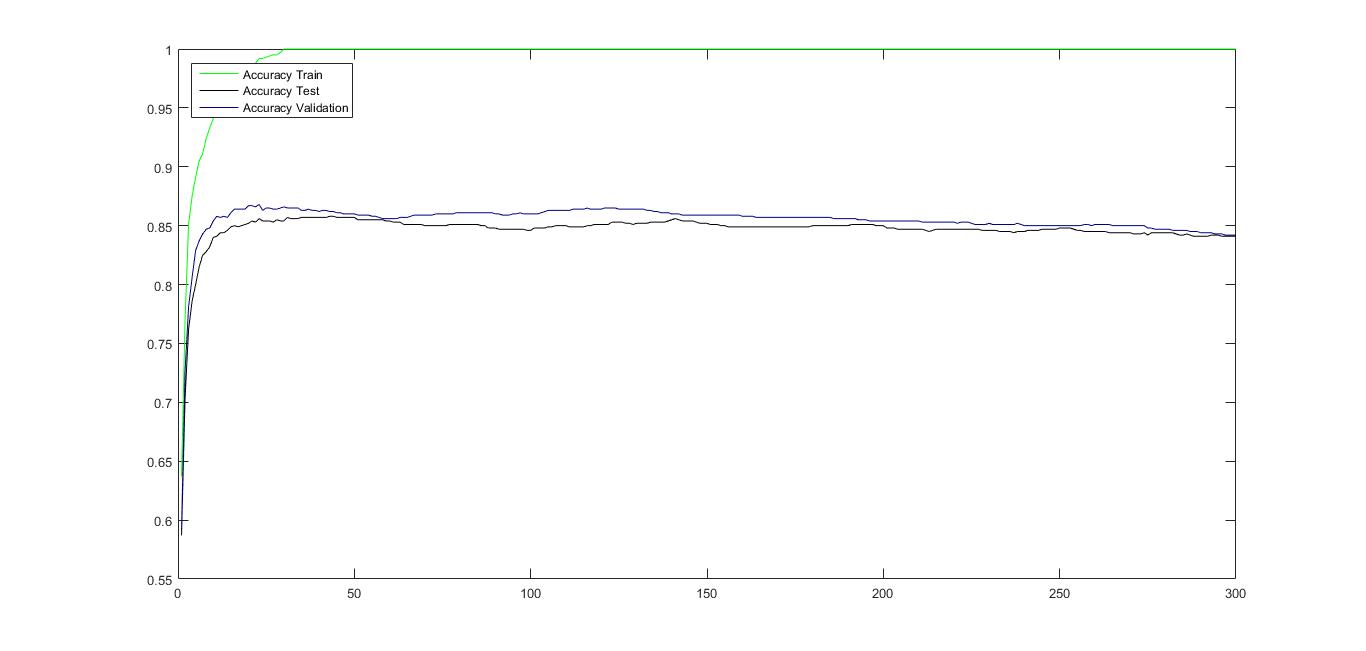


Figure - Accuracy Regularization = 0.01 Figure - MSE Regularization = 0.01

R=0.001

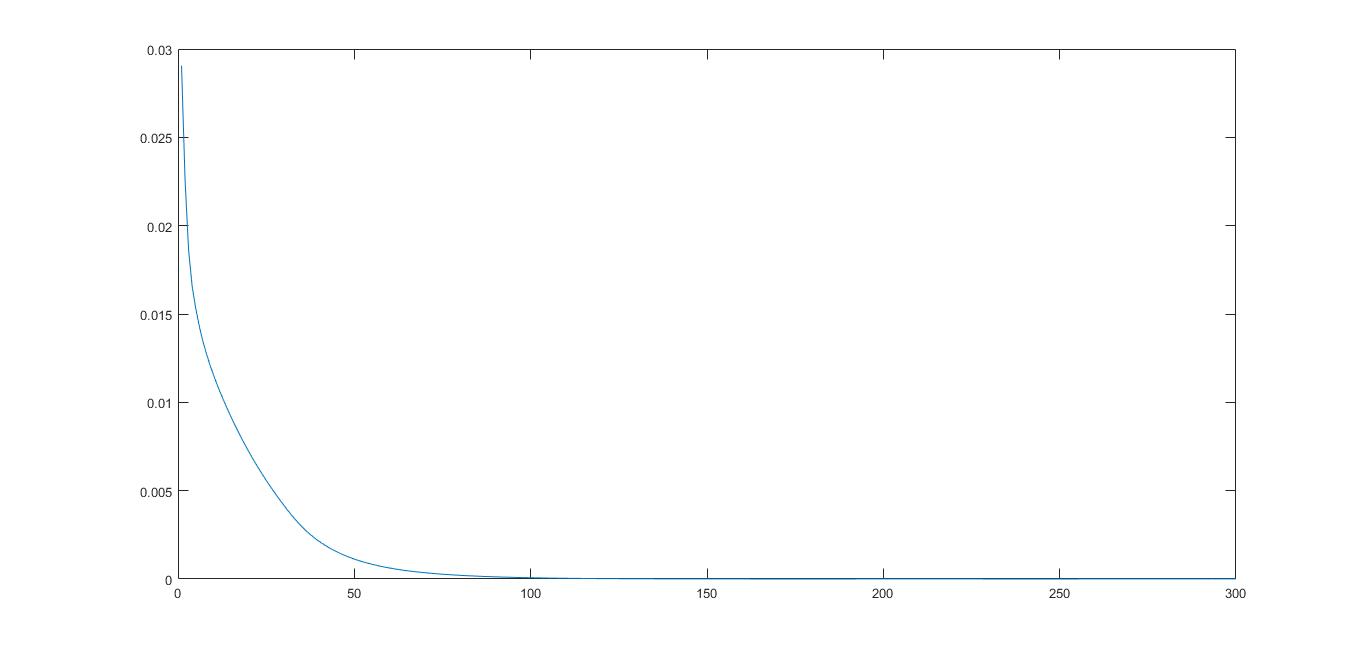


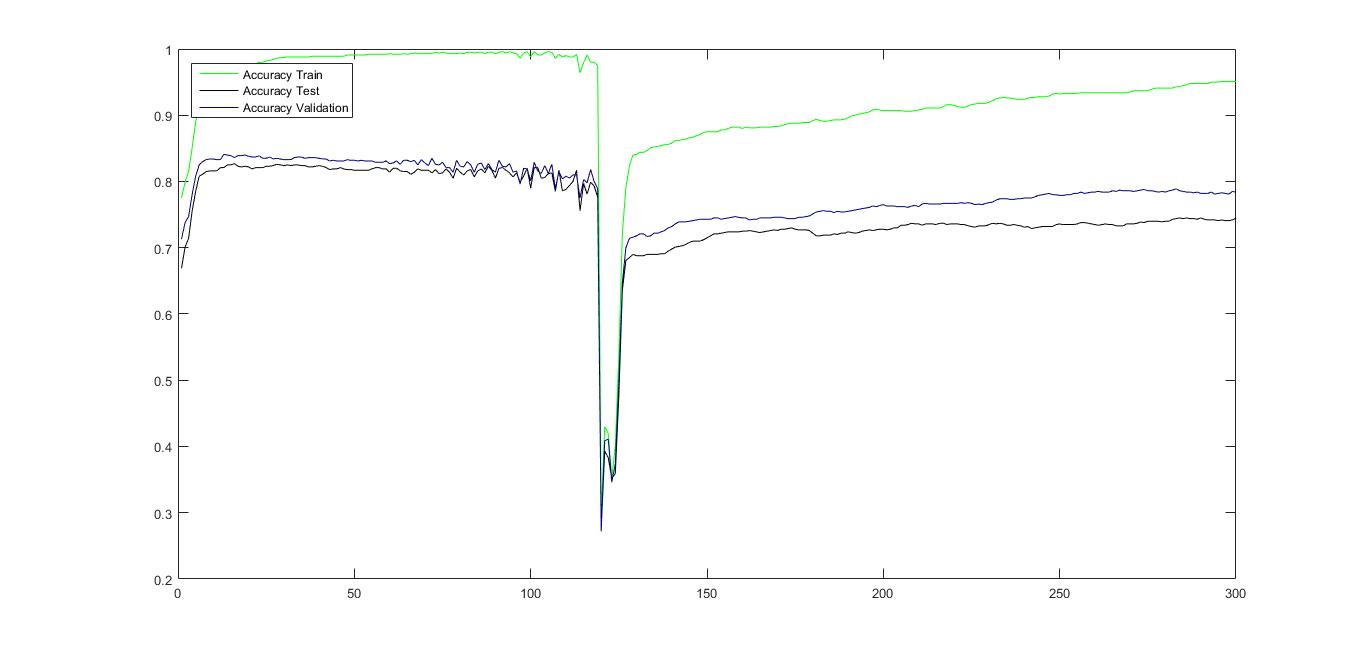
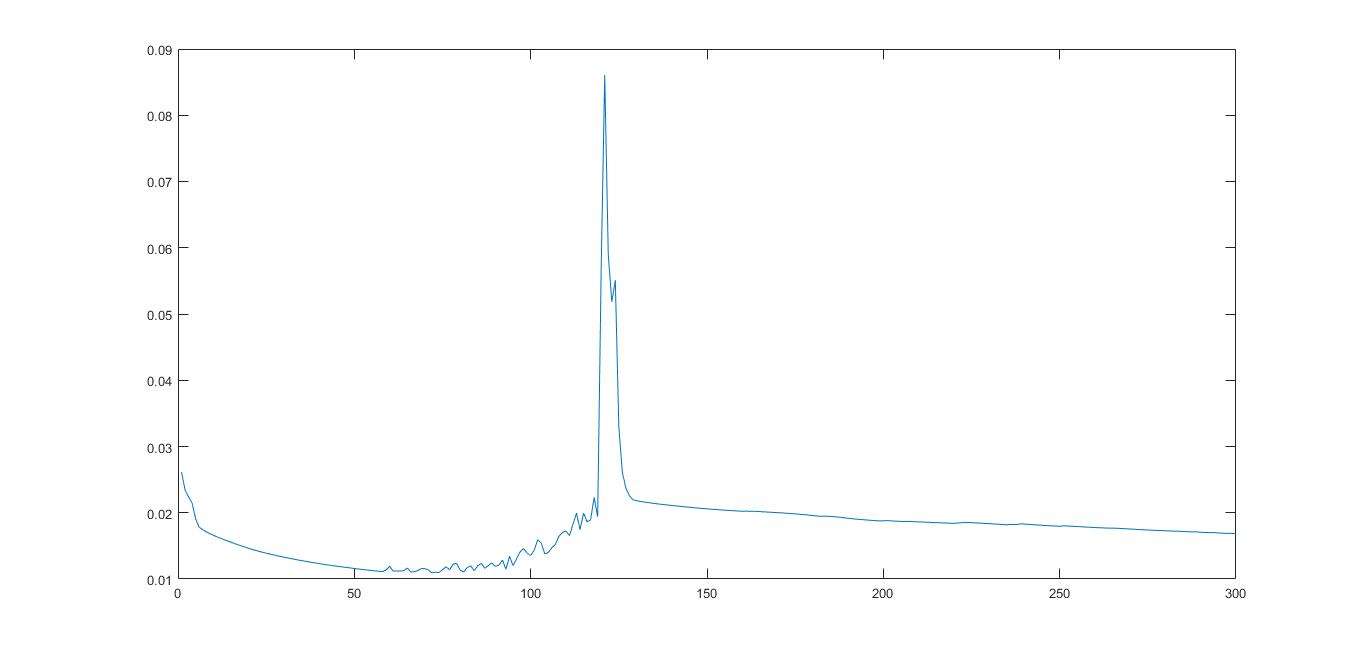
Figure - Accuracy Regularization = 0.001 Figure - MSE Regularization =0.001

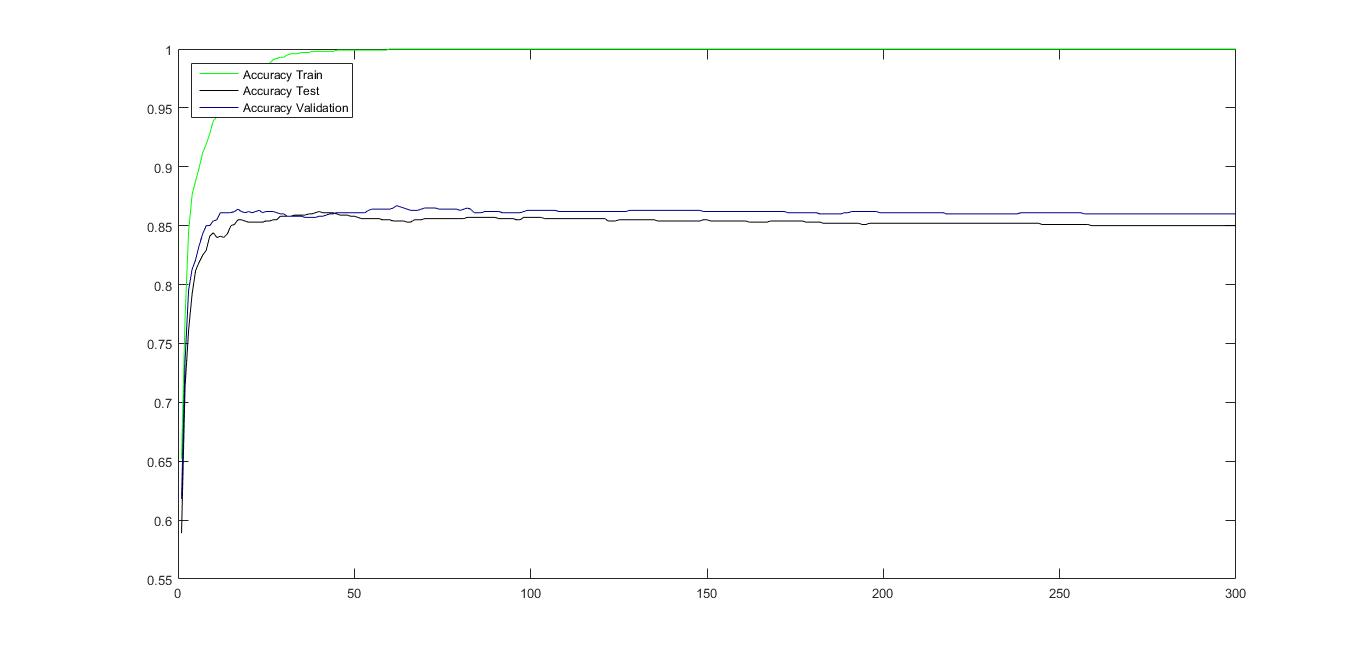
Table 2 Regularization

|  |  |  |
| --- | --- | --- |
| مقدرا ضریب منتظم سازی | دقت بر روی نمونه های تست | میزان خطای مجموع مربعات |
| R = 1 | 12.8 درصد | 0.317 |
| R = 0.5 | 16.5 درصد | 0.031796 |
| R = 0.01 | 74 درصد | 0.007 |
| R = 0.001 | 84 درصد | تقریبا صفر |

این ضریب هم رفتاری مطابق انتظار دارد. یعنی مقدار زیاد ضریب متنطم سازی باعث میشود شبکه چیزی یاد نگیرید 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 - Accuracy tan hyperbolic Figure - MSE tan hyperbolic

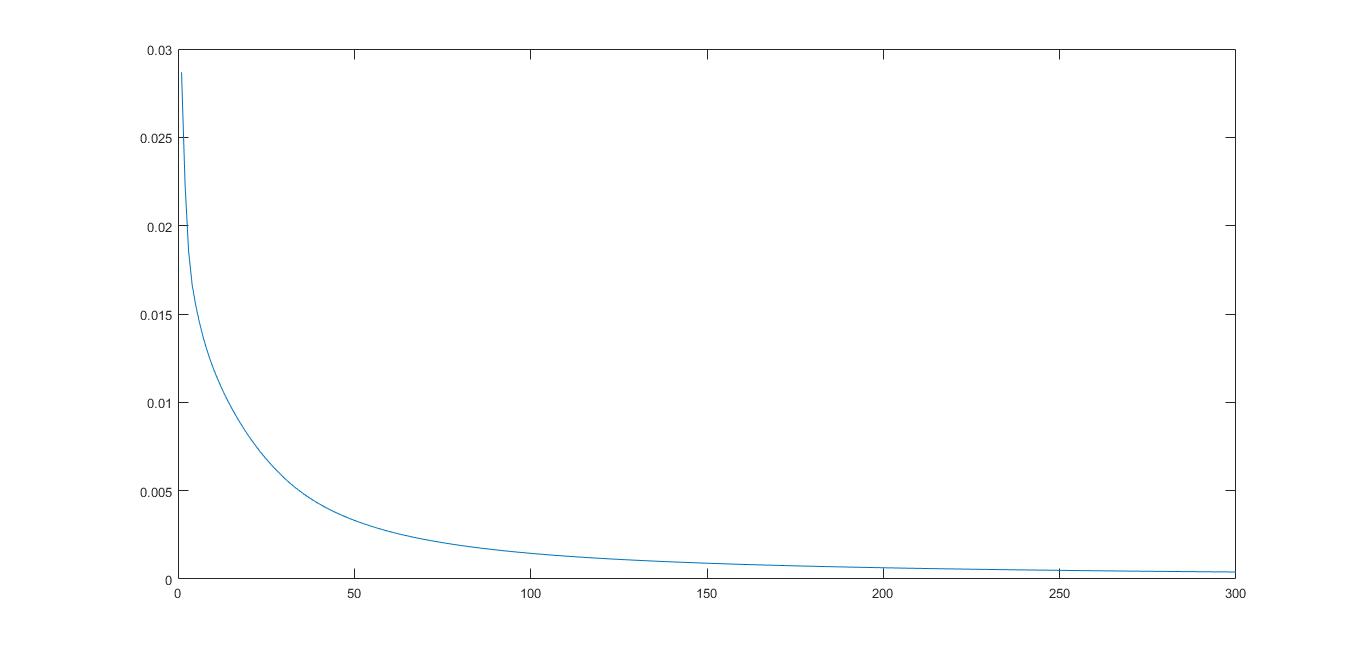


Figure - Accuracy Sigmoid Figure - MSE Sigmoid

Table 3 - Activation Function

|  |  |  |
| --- | --- | --- |
| تابع لجستیک | دقت بر روی نمونه های تست | میزان خطای مجموع مربعات |
| تانژانت هایپربولیک | 78.9 درصد | 0.00862 |
| سیگمویید | 85 درصد | 0.0003965 |

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

# تعداد لایه ها

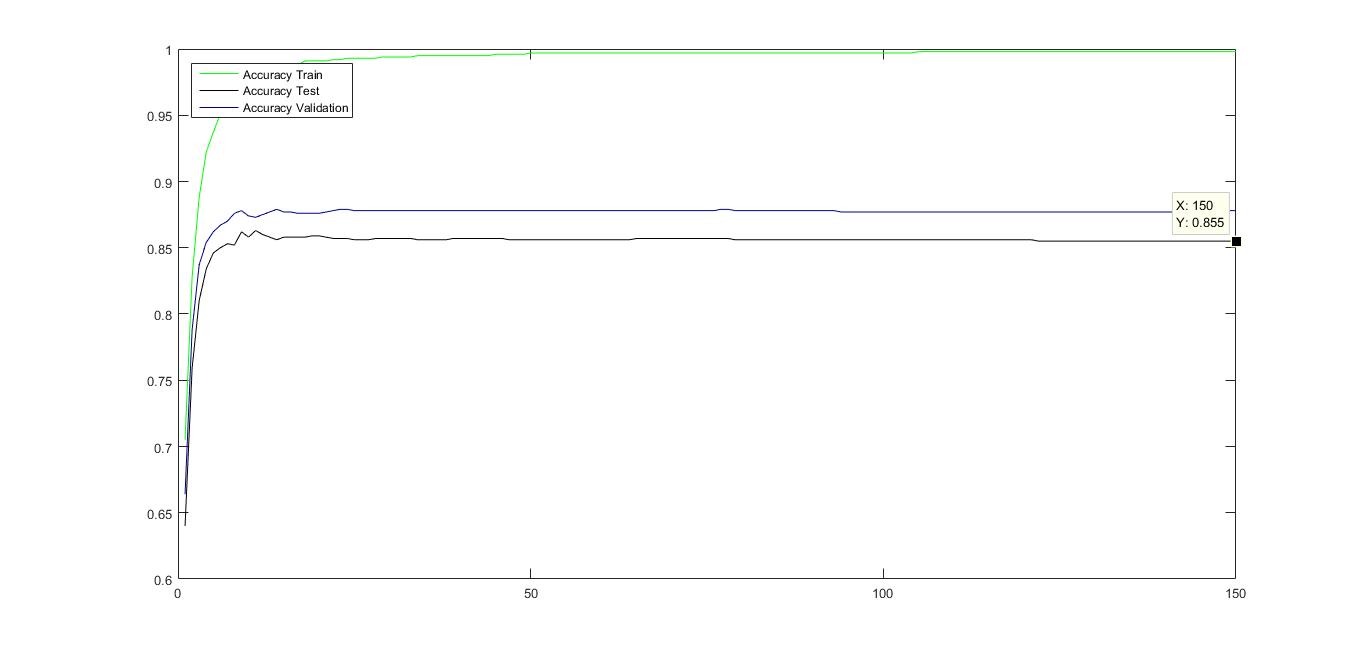
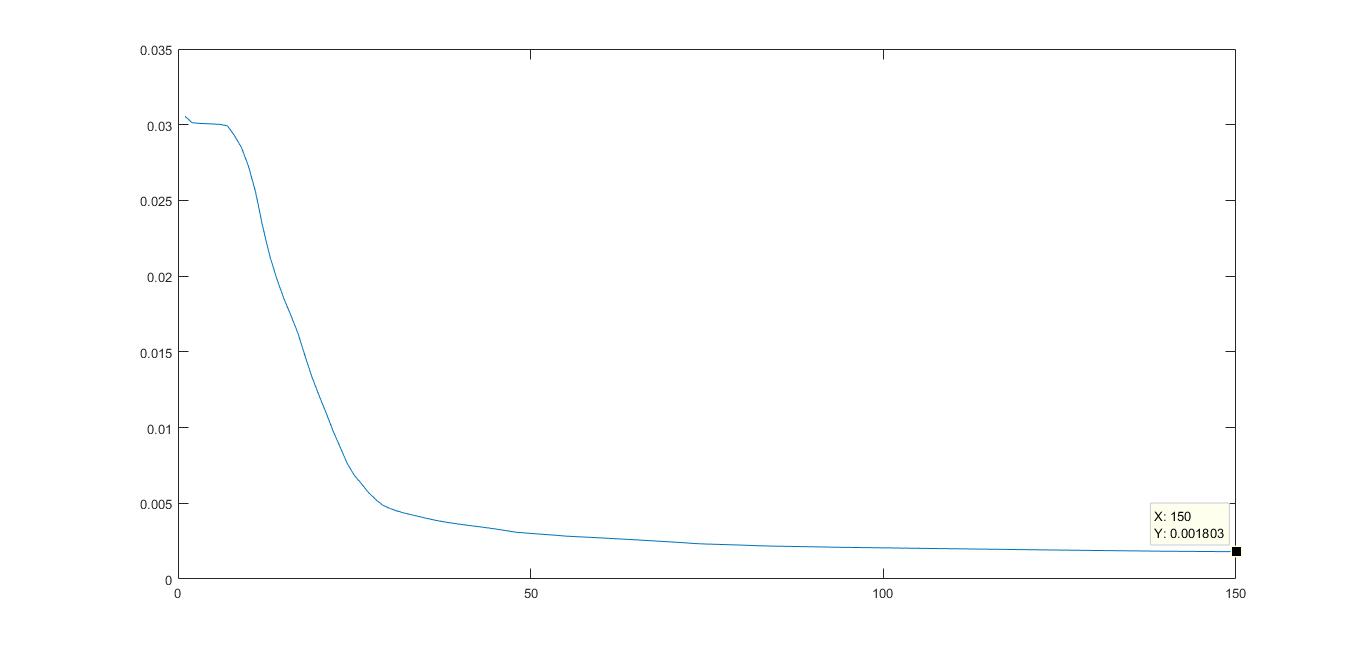
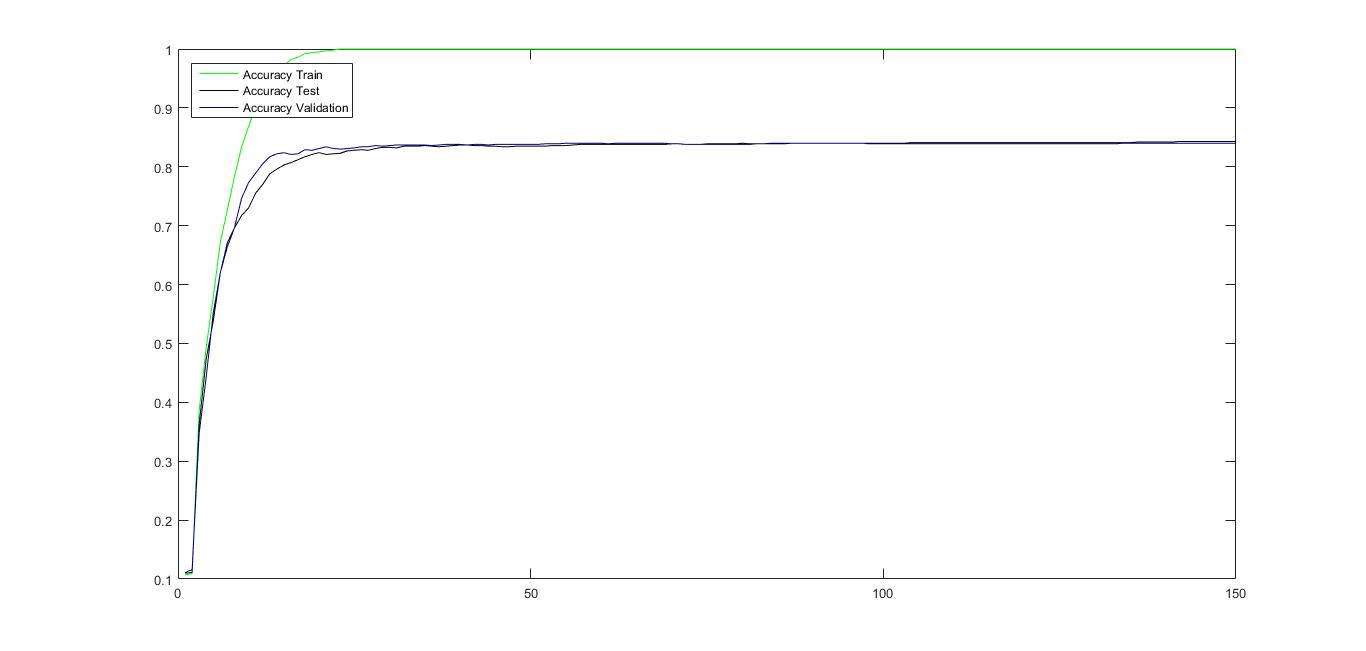
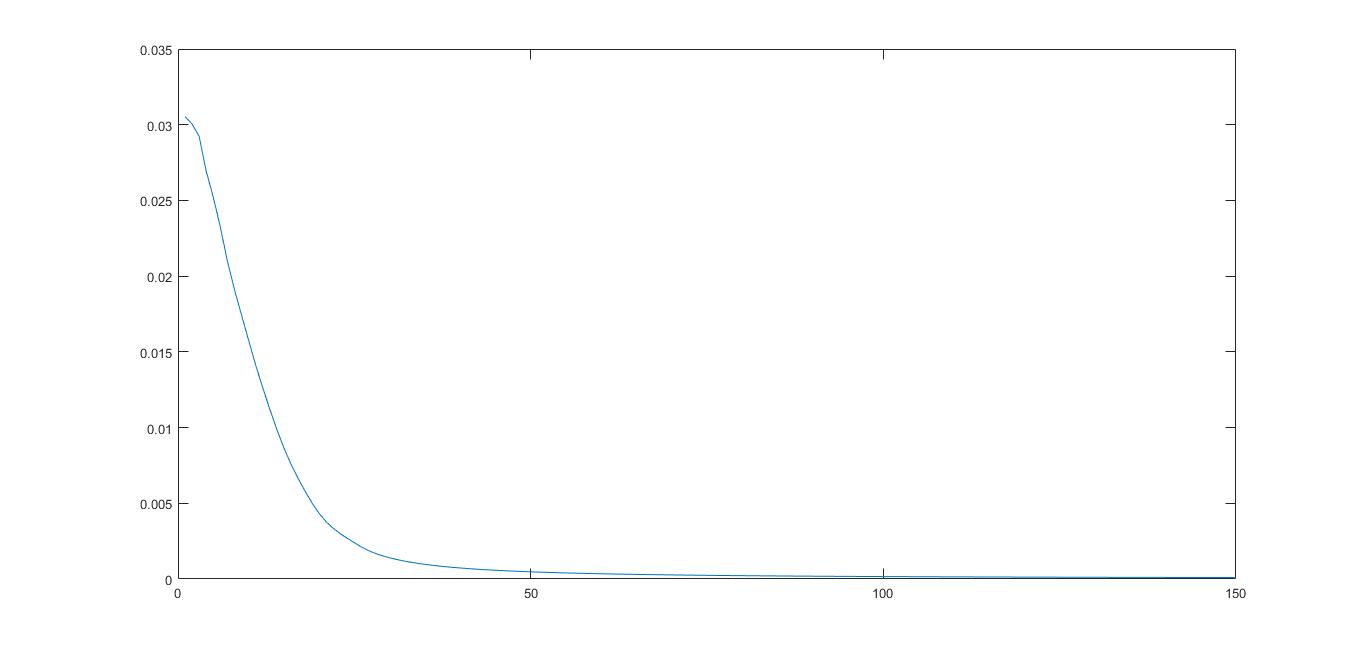


Figure - Accuracy 3 Layer 784:50:10 Figure - MSE 3 Layer 784:50:10



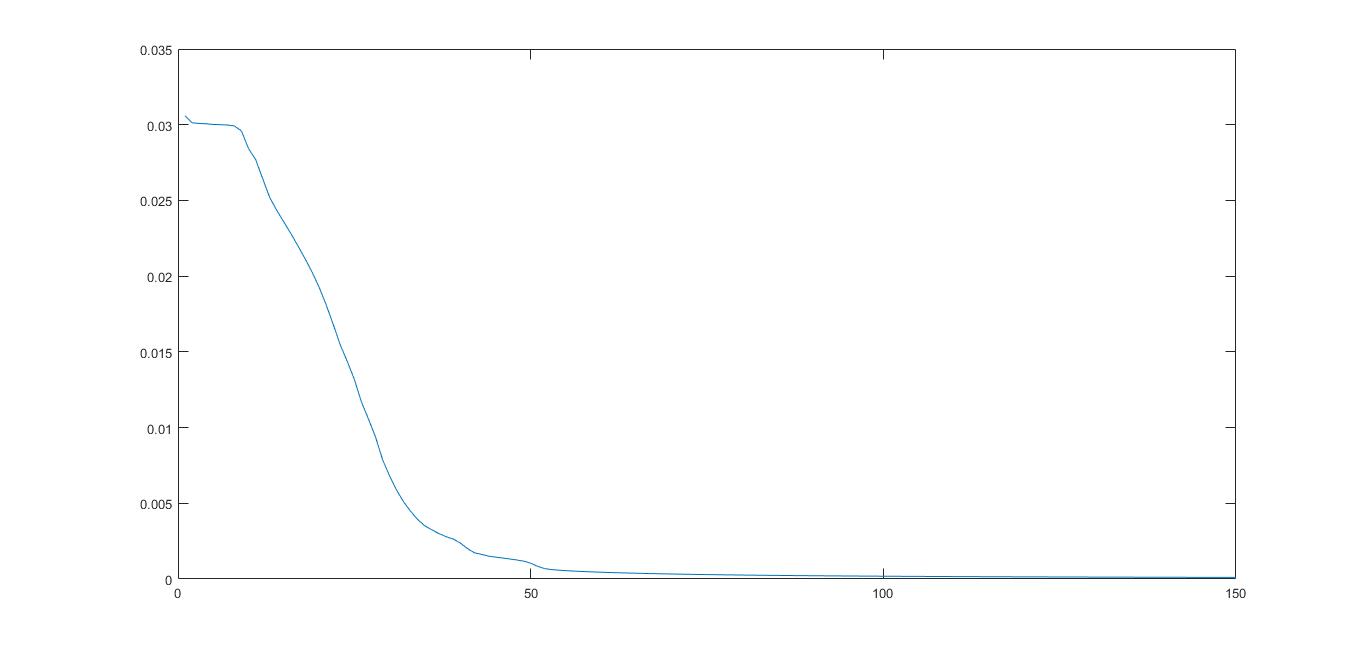
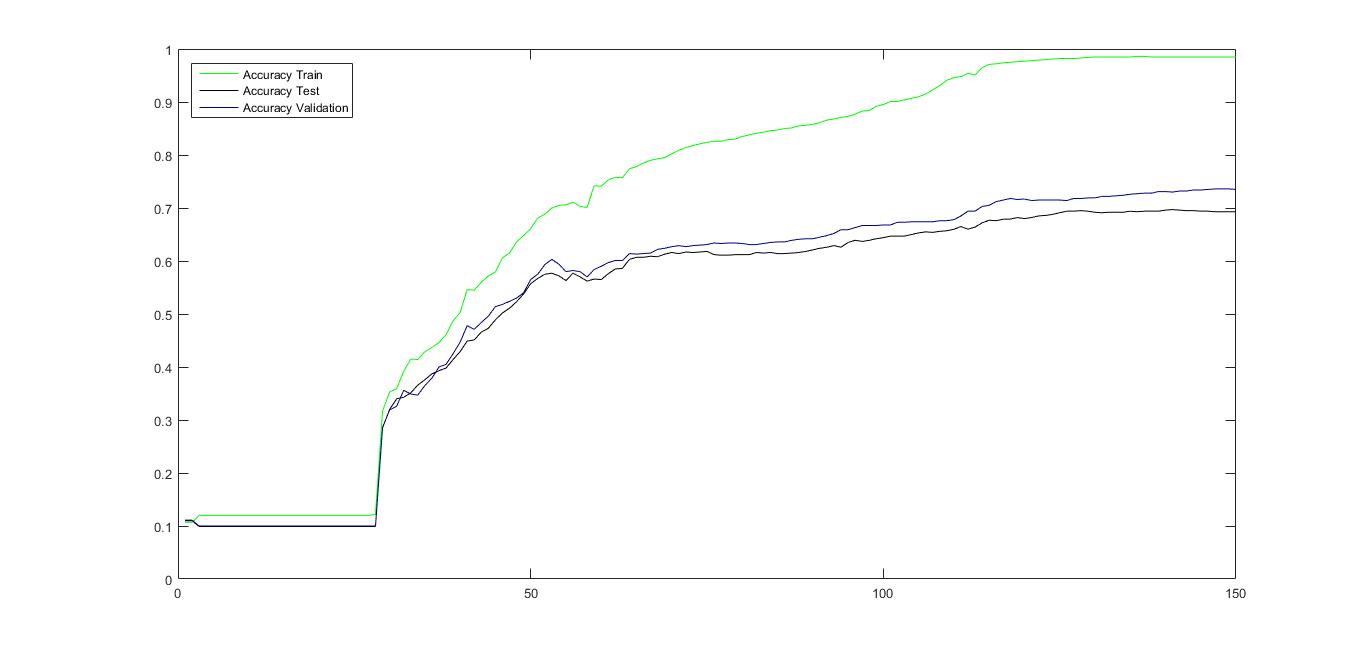
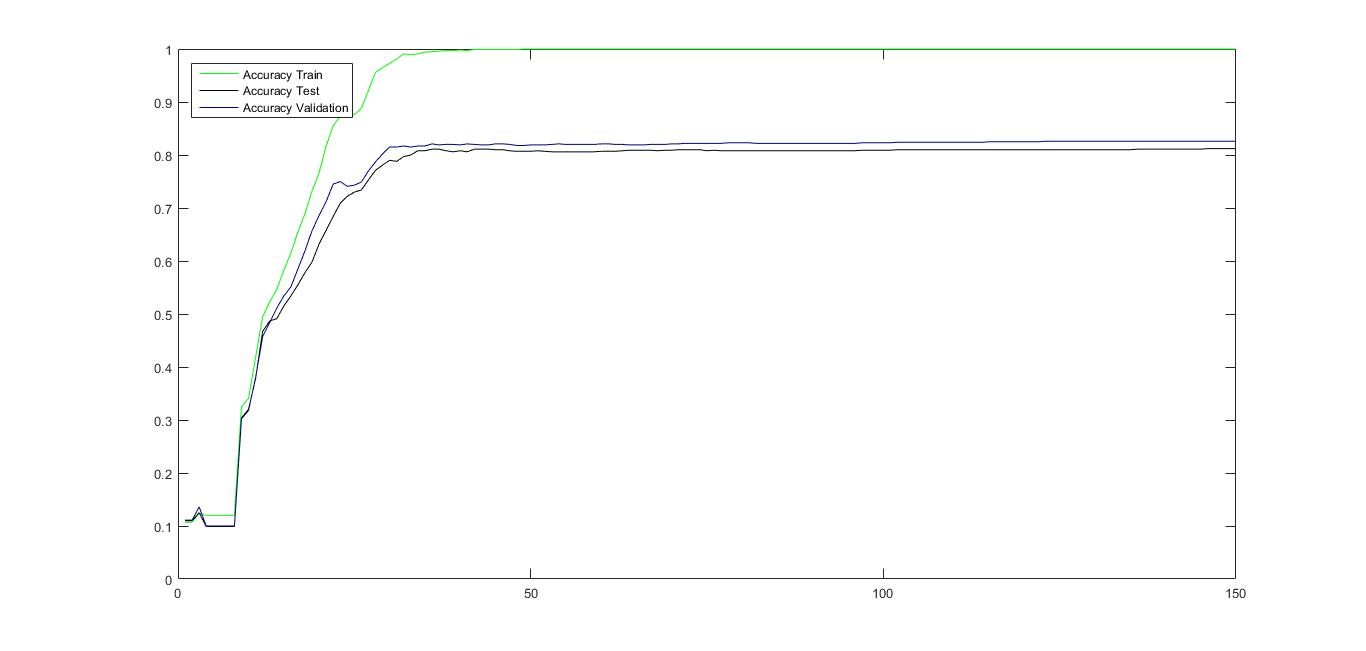
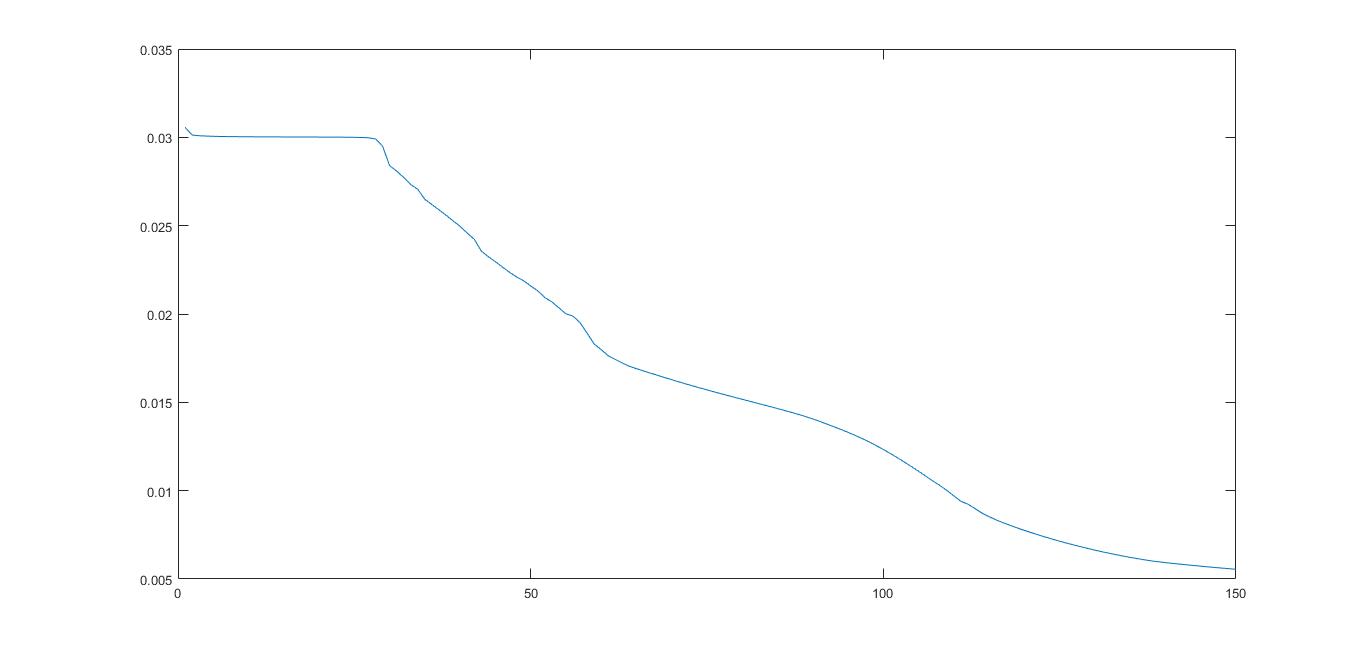
Figure - Accuracy 4 Layer 784:100:50:10 Figure 34 - MSE 4 Layer 784:100:50:10

Figure - Accuracy 5 Layer 784:150:100:50:10 Figure 36 - MSE 5 Layer 784:150:100:50:10



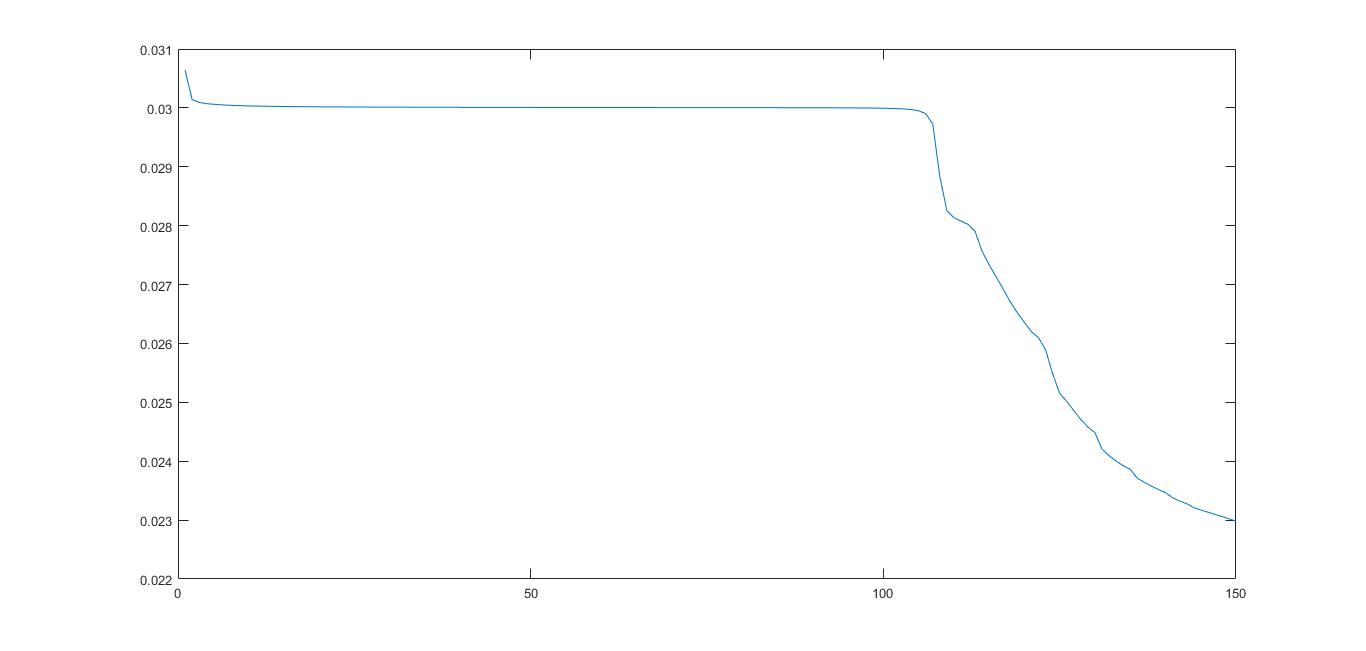
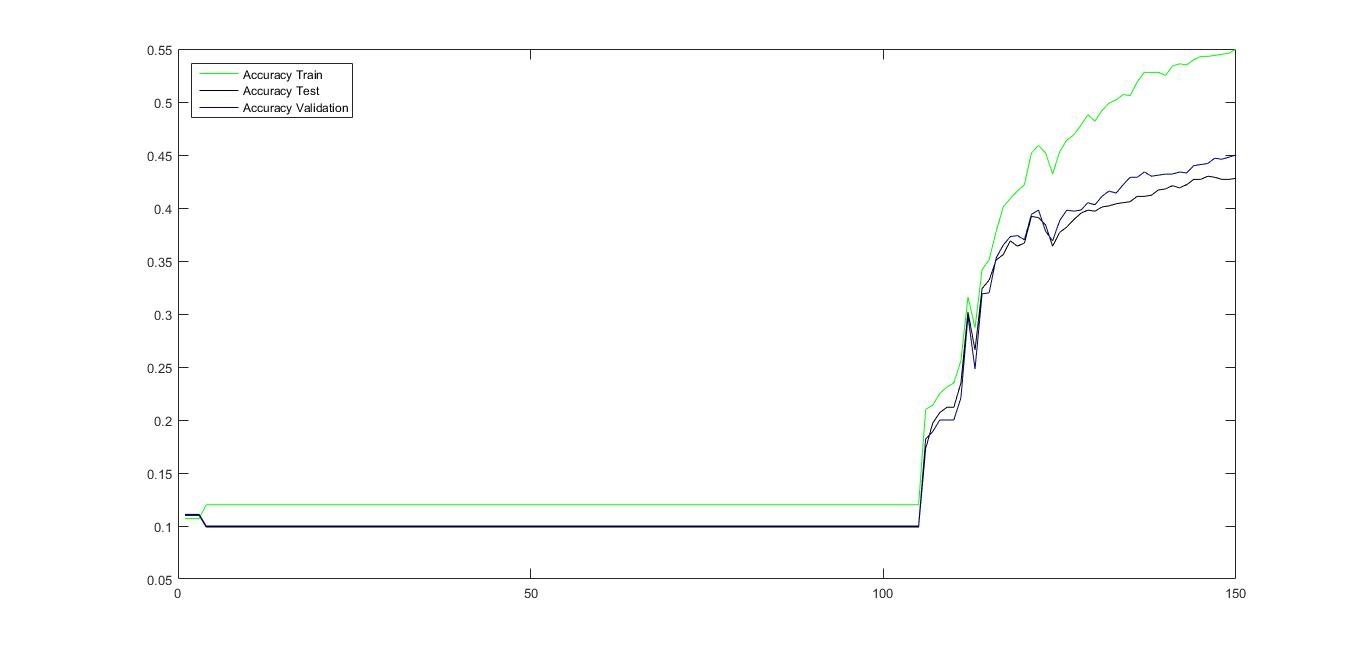
Figure - Accuracy 6 Layer 784:200:150:100:50:10 Figure - MSE 6 Layer 784:200:150:100:50:10

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

Table 4 - Compare layer size

|  |  |  |
| --- | --- | --- |
| تعداد لایه ها | دقت بر روی نمونه های تست | میزان خطای مجموع مربعات |
| 7 | 42.8 درصد | 0.02298 |
| 6 | 69.3 | 0.005549 |
| 5 | 80 درصد | 0.001803 |
| 4 | 82.6 درصد | 0.0002253 |
| 3 | 85.5 درصد | 0.004295 |

با افزایش تعداد لایه ها باعث کاهش سرعت همگرایی میشویم، اما این کار باعث می شود تعداد پارامتر های آزاد افزایش یابد و امکان وقوع پیش پوشش را بیشتر می کند. اما بر طبق این اصل که اگر بعد داده ها نسبت به تعداد آنها بیشتر باشد باعث می شود 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:

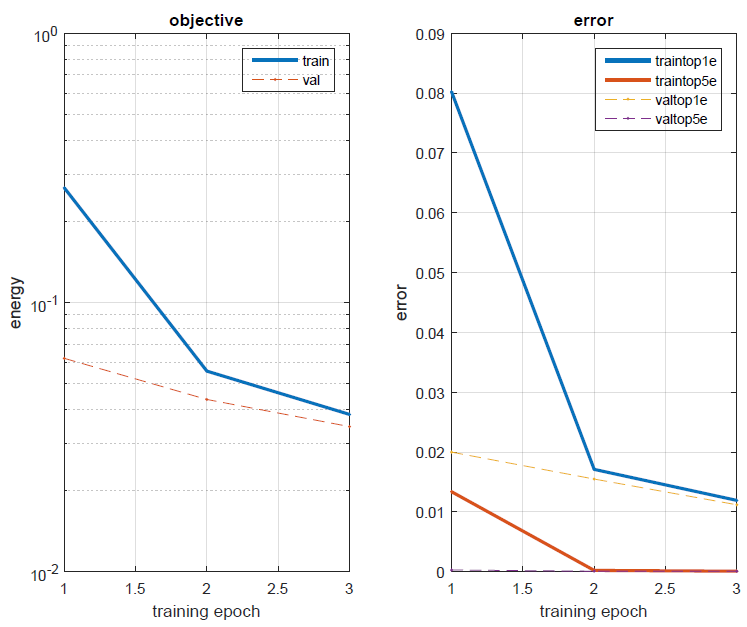


Figure - Accuracy CNN

همانگونه که مشاهده میکنید با استفاده از شبکه های عصبی کانولوشن که یک نوع شبکه عمیق می باشد، ما به دقت 99.9 درصد رسیده ایم.(خط زرد رنگ)

شبکه طراحی شده برای این کار به صورت زیر است:

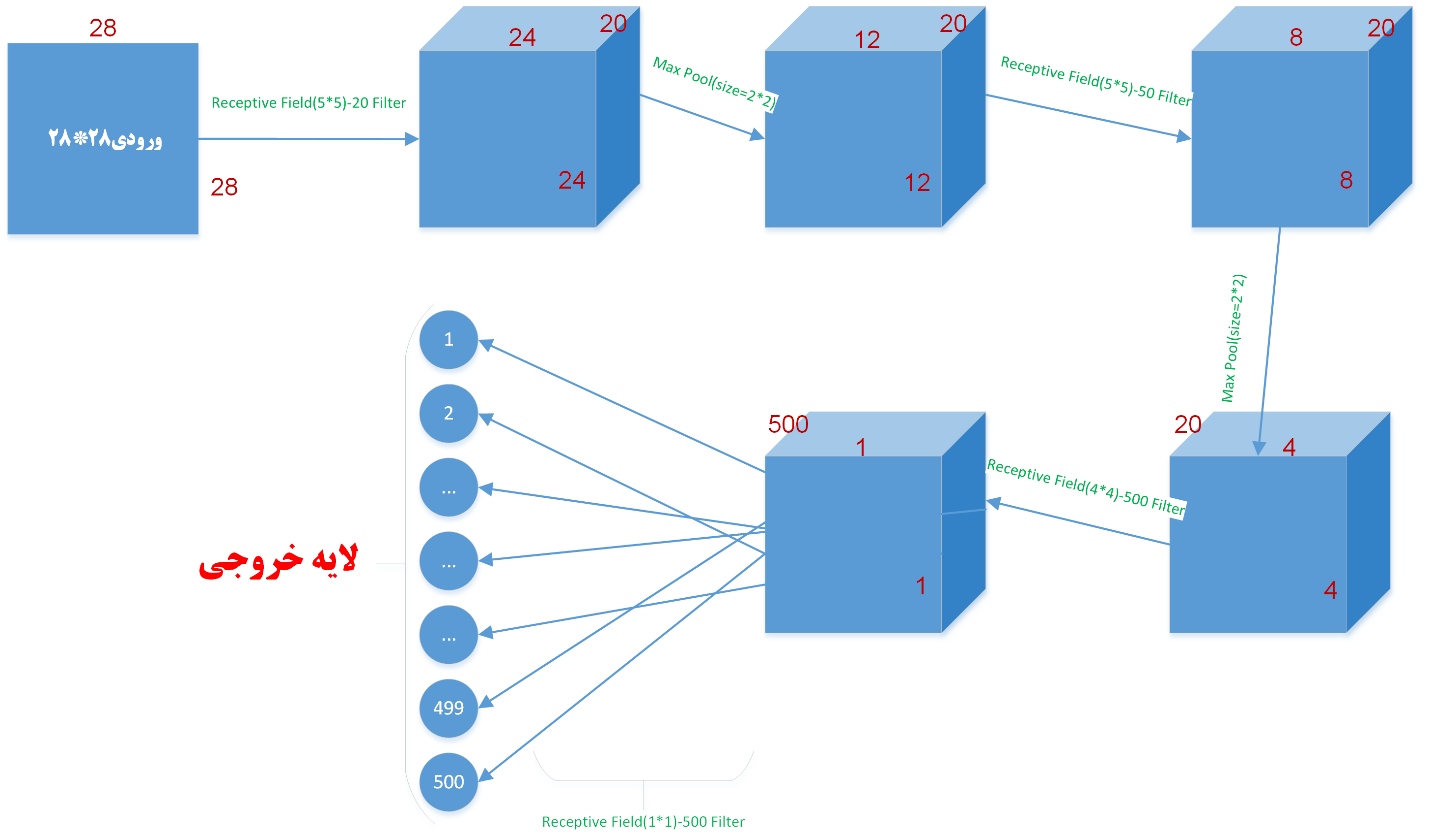


Figure - CNN Architecture

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 98.17 درصد بر روی داده های تست است و دقت شبکه عمیق برابر با 99.9 درصد بر روی داده های تست است.

# مراجع

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