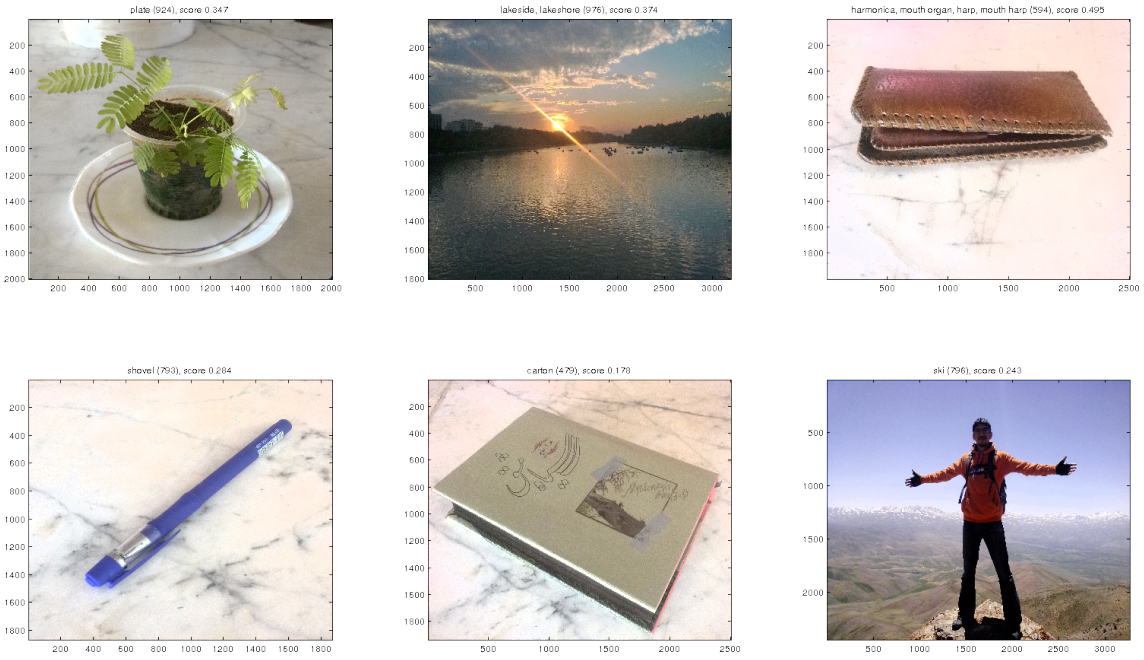
گزارش تمرین سوم
طبقه‌بندی تصاویر با استفاده از
Convolutional Neural Network و GoogleNet

داریوش حسن‌پور آده

۹۳۰۸۱۶۴

۱ قسمت ۱

بنده ۶ عکس از محیط اطراف گرفتم را به شبکه دادم و نتایج‌اش به صورت زیر بدست آمد.



که عکس اول (ردیف بالا، سمت راست) مربوط به «کیف پول» بنده می‌باشد که شبکه «سازدهنی» طبقه بندی است، عکس دوم مربوط که غروب خورشید که روی پل سی‌وسی پل گرفته شده است، که شبکه «دریاچه» طبقه‌بندی کرده است. عکس سوم مربوط به گیاهی است که در یک داخل یک زیراستکانی قرار داده شده است، که شبکه «بشقاب» طبقه‌بندی کرده است، عکس چهارم (ردیف پایین، سمت راست) مربوط به عکس بنده در ارتفاعات کوه‌ها می‌باشد، که شبکه «آسمان» طبقه‌بندی کرده است، عکس چهارم مربوط که به یک کتاب شعر از

«هوشنگ ابتهاج» می‌باشد که شبکه «مقوا» طبقه‌بندی کرده است و آخر مربوط به یک عدد خودکار می‌باشد که شبکه «بیل» طبقه‌بندی کرده است. همان‌طور که می‌بینیم به جز عکس آخر(خودکار) در مابقی عکس‌ها آنچه که شبکه طبقه‌بندی کرده است با آنچه که عکس‌ها بودند چندان هم بی‌ربط نیست، که نشان از خوب کار کردن شبکه‌ی GoogleNet می‌باشد.

۲ قسمت ۲

در این قسمت عکس‌هایی که برای طبقه‌بندی در نظر گرفته است، عکس‌های MRI مربوط به «مغز»[۱] و «زانو»[۲] می‌باشد که از موتور جستجوگر گوگل جمع‌آوری شده‌اند. همان‌طور که خواسته شده است ٪۸۰ از داده به صورت تصادفی به عنوان داده‌های آموزشی و ٪۲۰ مابقی را به عنوان داده‌های تست در نظر گرفته شدند. نتایج هر ۳ قسمت تکلیف در شکل زیر آمده است. همان‌طور که در شکل بالا مشاهده می‌شود به ازای هر

```
Loading 2 classes' images....[DONE]
Building 1-NN model....[DONE]
Testing the model...[DONE]
Confusion Matrix:
              pred_1    pred_2
    real_1    17         1
    real_2     2        12

Loading CNN data...[DONE]
Extracting features....[DONE]
Training MLP....[DONE]
Testing model....[DONE]
Confusion Matrix:
              pred_1    pred_2
    real_1    19         0
    real_2     0        13

Reducing deminsions from 50176 -> 1000....[DONE]
Training MLP....[DONE]
Testing model....[DONE]
Confusion Matrix:
              pred_1    pred_2
    real_1    16         2
    real_2     3        11
```

زیرقسمت(یعنی آموزش و ساخت مدل توسط «نزدیک‌ترین همسایه[۳]»، استفاده از گوگل‌نت به عنوان استخراج کننده‌ی ویژگی و یک شبکه‌ی چندلایه و استفاده از PCA و یک شبکه‌ی چندلایه) داده‌های تست را با استفاده

[۱] ۶۲ عدد عکس
[۲] ۸۰ عدد عکس
[۳] K-NN where K = 1

از مدل بدست آمده تست کرده و Confusion Matrix آن را رسم کرده‌ایم. برای اینکه بتوانیم نتایج حاصل از PCA را با نتایج حاصل از گوگل‌نت مقایسه کنیم به تبعیت از ساختار شبکه‌ی گوگل‌نت عکس‌ها را از یک بردار ۵۰،۱۷۶ به یک بردار ۱۰۰۰ تایی کاهش دادیم و سپس توسط یک شبکه‌ی ۱ × ۳۰ × ۱۰۰۰ آموزش دادیم(همین ترکیب شبکه برای قسمت ۲.۲ تکلیف نیز در نظر گرفته شده است.). همان‌طور که می‌بینیم شبکه ای که با توسط گوگل‌نت کاهش بعد داده شده است بدون خطا همه‌ی داده‌های تست را به درستی طبقه‌بندی کرده است در حالی که هم در نزدیک‌ترین همسایه و هم در PCA خطای طبقه‌بندی مشاهده می‌شود؛ که نشان می‌دهد گوگل‌نت می‌تواند کاهش‌بعد دهنده‌ی خوبی باشد.

خروجی شبکه‌ی چندلایه در هردو قسمت ۲.۱ و ۲.۲ یک نورون بوده که برای کلاس یکی از دسته عکس ها مقدار هدف ۰ درنظر گرفته شده و برای دیگری ۱، در خروجی شبکه اگر بیشتر از ۰.۵ باشد ۱ در نظر گرفته می‌شود و اگر کمتر از ۰.۵ باشد ۰ در نظر گرفته می‌شود. در «نزدیک‌ترین همسایه» ٪۹ خطا، در «کاهش بعد با گوگل‌نت» ٪۰ خطا و در «کاهش بعد با PCA» ٪۱۵ خطا داشته‌ایم، همان‌طور که می‌بینیم «کاهش‌بعد توسط گوگل‌نت و طبقه‌بندی توسط MLP» حتی از «نزدیک‌ترین همسایه» نیز بهتر عمل کرده است.

۳ قسمت ۳

در این قسمت بنده عکس‌ها را دانلود کرده و برچسب‌های عکس‌های مرتبط با هر یک از عکس‌ها را استخراج کرده و سعی در ایجاد شبکه‌ای کانولشنی که بتواند بروی داده‌ها یادگیری انجام دهد کردم، متاسفانه نسخه‌ی beta17 کتابخانه‌ی MatConvnet مشکل داشت و بخاطر این موضوع بود که در نسخه‌ی ابتدایی این تکلیف که بنده قبل از ۱۳ دی آپلود کردم مشکل عدم توانایی ایجاد شبکه‌ای جهت یادگیری داده‌ها را گزارش کردم[۴] که بعد از تغییر نسخه‌ی کتابخانه به beta16 مشکل گزارش شده برطرف شد؛ از آنجایی که هیچ‌گونه مستند رسمی جهت چگونگی آموزش شبکه توسط این کتابخانه منتشر نشده است، به ناچار از مهندسی معکوس مثال های موجود در فایل‌های کتابخانه استفاده شد تا بتوانم شبکه را بنویسم. ظاهرا به باید داده‌های ورودی شبکه در قابل یک ساختار با فرمت مشخص به تابع یادگیری cnn_train[۵] ارسال شود که بنده مشابه کدهای آنچه که در مثال برای بارگذاری داده‌های mnist آورده شده است برای بارگذاری داده‌های Jaffe کدهایی نوشتم.

در حالت خلاصه بنده از تابع cnn_train موجود در میان مثال‌های کتابخانه جهت آموزش شبکه‌ی کانولوشن استفاده کردم، تابع getImdb که وظیفه‌ی بارگذاری داده‌ها به فرمتی که سازگار با تابع cnn_train باشد را با مهندسی معکوس مثال‌های موجود در کتابخانه نوشتم و در نهایت یک اسکریپت جهت اجرای قسمت سوم نوشته شده است.

بنده با استفاده از ۳ لایه‌ی کانولشنی و ۲ عدد لایه‌ی max-pooling توانستم در نهایت به دقت ٪۹۸ بروی داده‌های ارزیابی[۶] با معیار Top-1 برسم و با معیار Top-5 به دقت ٪۱۰۰ رسیدم و برای داده‌های آموزشی میزان دقت با هر دو معیار ٪۱۰۰ بوده است، البته لازم به ذکر است که معیار Top-5 در این تکلیف با این دیتاست

[۴] علت بارگذاری زودتر این تکلیف بخاطر این بود که دکتر صفایانی در ابتدا تاریخ ۱۳ را به عنوان آخرین مهلت بارگذاری تکلیف اعلام کرده بودند و بعد از چند روز بعد از بارگذاری تکلیف توسط بنده اعلام شد که مهلت ارائه‌ی تکلیف به ۳۰ دی ماه تغییر یافت!

[۵] که این تابع را نیز مثال‌های موجود در کتابخانه استخراج کردم در کنار کدها ارسال نموده‌ام.

[۶]

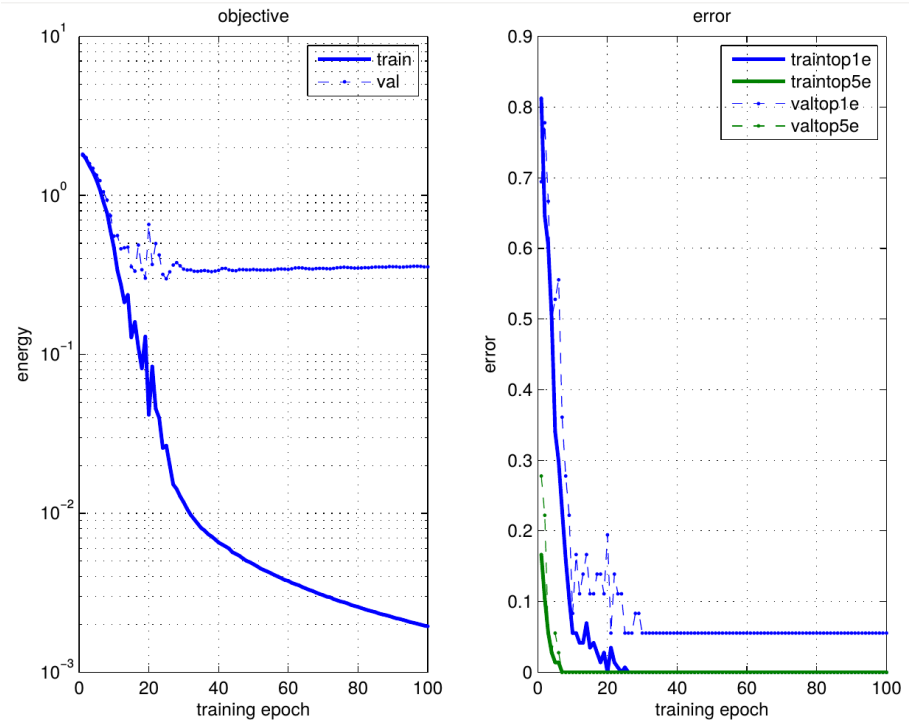
خاص که کلا ۶ کلاس دارد به درد نمی‌خورد و اطلاع بدرد بخوری از میزان خوبی عملکرد شبکه به ما نمی دهد(زیرا کلا ۶ کلاس داریم.). بطور خلاصه توضیح در مورد دو میعار گفته شده(یعنی Top-1 و Top-5) می‌توان گفت که برای اولین بار توسط .Krizhevskym et al[۱] ارائه شد. معیار Top-1 به این معنی است که در صورتی که کلاس واقعی نمونه‌ای با کلاسی که خروجی شبکه به عنوان بیشترین و محتمل‌ترین کلاس آن نمونه پیش ببینی می‌کند برابر باشد می‌گوییم که شبکه درست تشخیص داده است و یکی شمارنده تشخیص درست اضافه میکنیم و در نهایت آن شمارنده را به تعداد نمونه‌ها تقسیم می‌کنیم که میزان دقت شبکه می‌شود؛ معیار Top-5 نیز همانند Top-1 می‌باشد با این تفاوت که در صورت که کلاس نمونه جز ۵ محتمل‌ترین کلاسی که شبکه برای آن نمونه تشخیص داده باشد، باشد آنگاه به شمارنده‌ی صحت شبکه یکی اضافه کرده و در نهایت به تعداد کل تقسیم می‌کنیم.

برای آموزش شبکه عکس‌های مرتبط با هر کلاس را که توسط پوشه‌ها جدا شده‌اند بارگذاری میکنیم و به طور تصادفی ٪۸۰ از این داده‌ها را به عنوان داده‌های آموزشی و ٪۲۰ مابقی به عنوان داده‌های تست انتخاب می‌کنیم به شبکه می‌دهیم. برای آموزش شبکه از تکنیک mini-batch با سایز ۴۰ استفاده کردم. تعداد اپوک‌ها ۱۰۰ و ضریب یادگیری ۰.۰۰۱ در نظر گرفته شده است. با استفاده از داده‌های آموزشی و تست استخراج شده شبکه را ۵ بار آموزش داده و در بعد از اتمام هر سری آموزش نتایج را ذخیره کرده و دوباره به آموزش شبکه با وزن‌دهی‌های اولیه جدید می‌پردازیم، این کار را ۵ بار طبق دستورالعمل ارائه شده در تکلیف انجام می‌دهیم. نتایج اجرای هر سری را در محل‌های جداگانه ذخیره میکنیم که در در شکل‌های ۱ـ۵ آمده است. همانطور که گفته شد معیار Top-5 برای داده‌های آموزشی این تکلیف که کلا ۶ کلاس دارد معیار به درد بخوری نیست لذا از اینجا به بعد منظور از «میزان دقت» همان میزان دقت با استفاده از معیار Top-1 می‌باشد. همان‌طور که مشاهده می‌شود شبکه اجرای خوبی بروی داده‌ها دارد و کمترین دقت ٪۹۴.۵ و بیشترین دقت ٪۱۰۰ را داشته است. در شکل‌های ۶ و ۷ میانگین این ۵ اجرا را در نموداری آورده شده است که بطور میانگین شبکه در نهایت دقت ٪۹۷.۳ را میزان انرژی ۰.۰۱ بروی داده‌های آموزشی و ۰.۱۳ بروی داده‌های تست دارد.

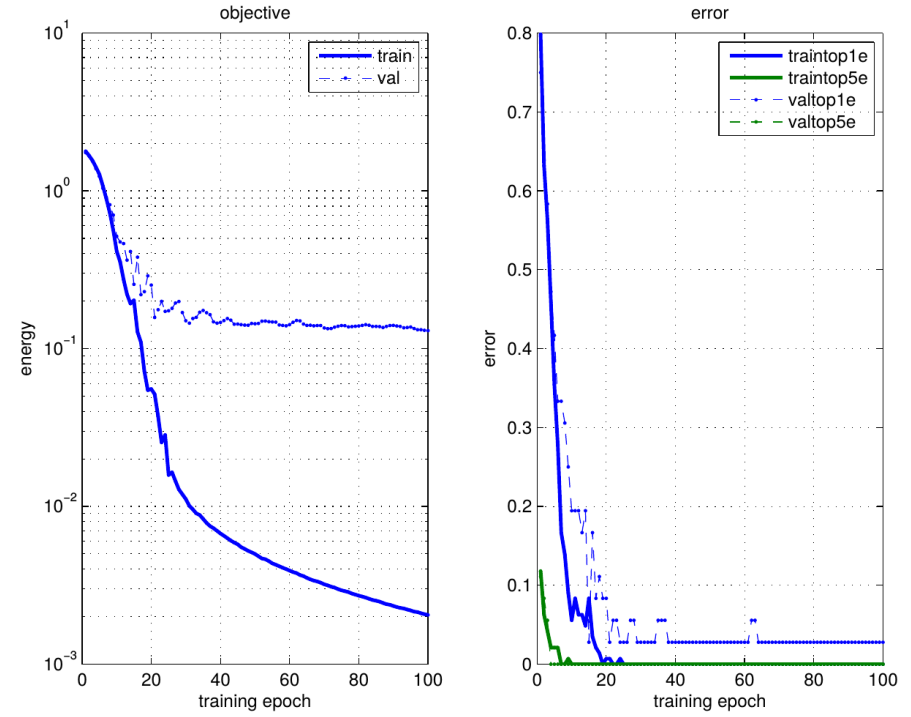
همان‌طور که مشاهده می‌کنیم نتایج اجرای شبکه‌ی کانولوشن رضایت بخش بوده است و در نتایج با وجود اینکه افت و خیزی مشاهده می‌کنیم ولی در نهایت به یک دقت خوبی همگرا می‌شود. داده‌های نتایج و شبکه‌ی هر اجرا به همرا کدها ارسال گردیده است.

مراجع

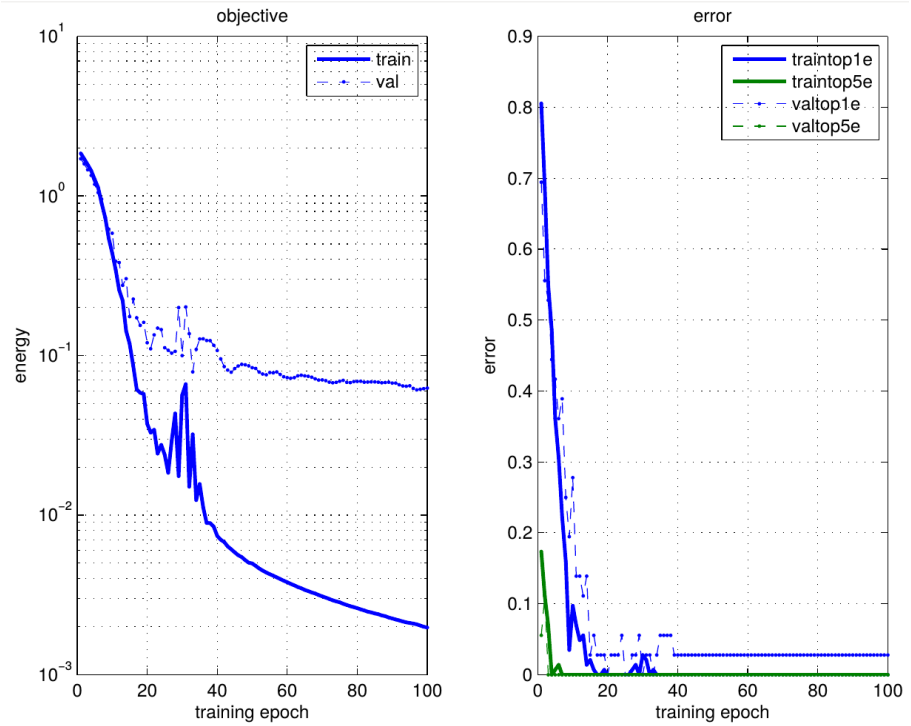
[1] A.Krizhevsky, I.Sutskever, and G.E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp.1097–1105, 2012.



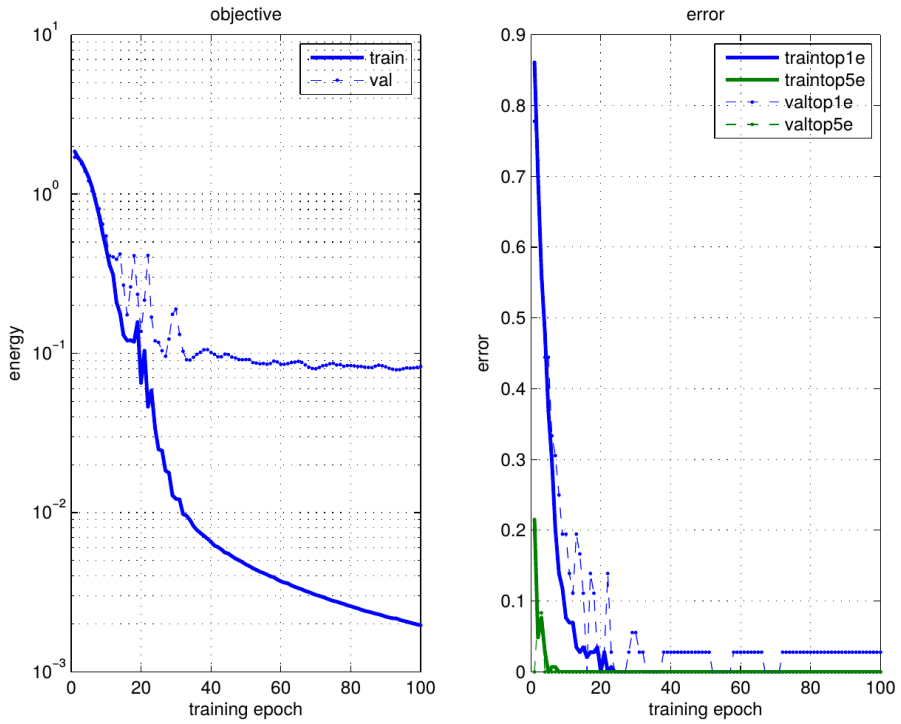
شکل۱: نتیجه‌ی اجرای ۵/۱، دقت بروی داده‌های تست: ٪۹۴.۵ انرژی شبکه بروی داده‌های تست: ۰.۳۵



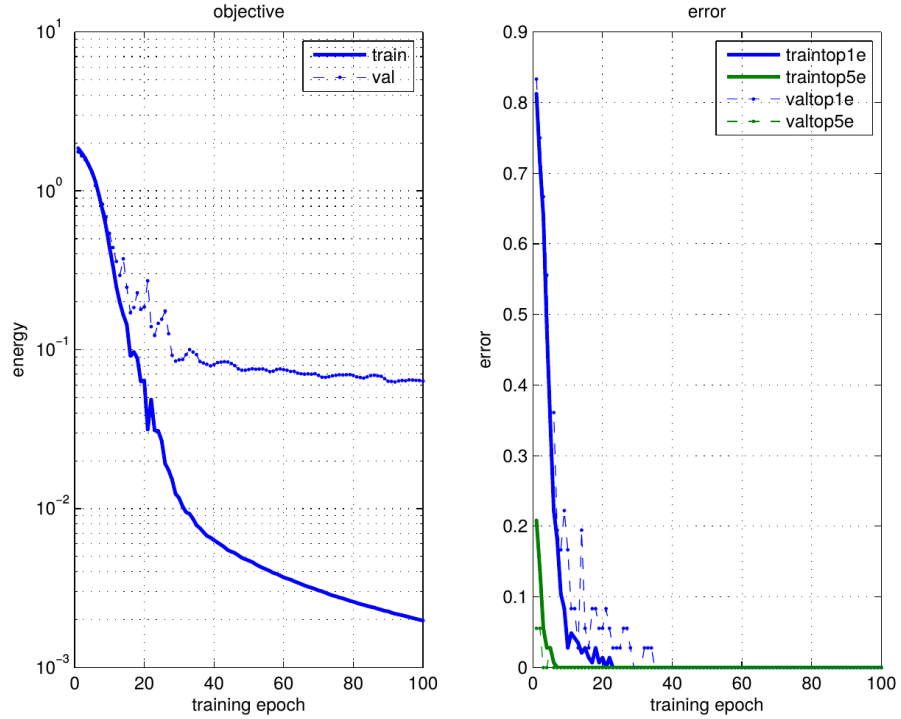
شکل۲: نتیجه‌ی اجرای ۵/۲، دقت بروی داده‌های تست: ٪۹۷.۲ انرژی شبکه بروی داده‌های تست: ۰.۱۲



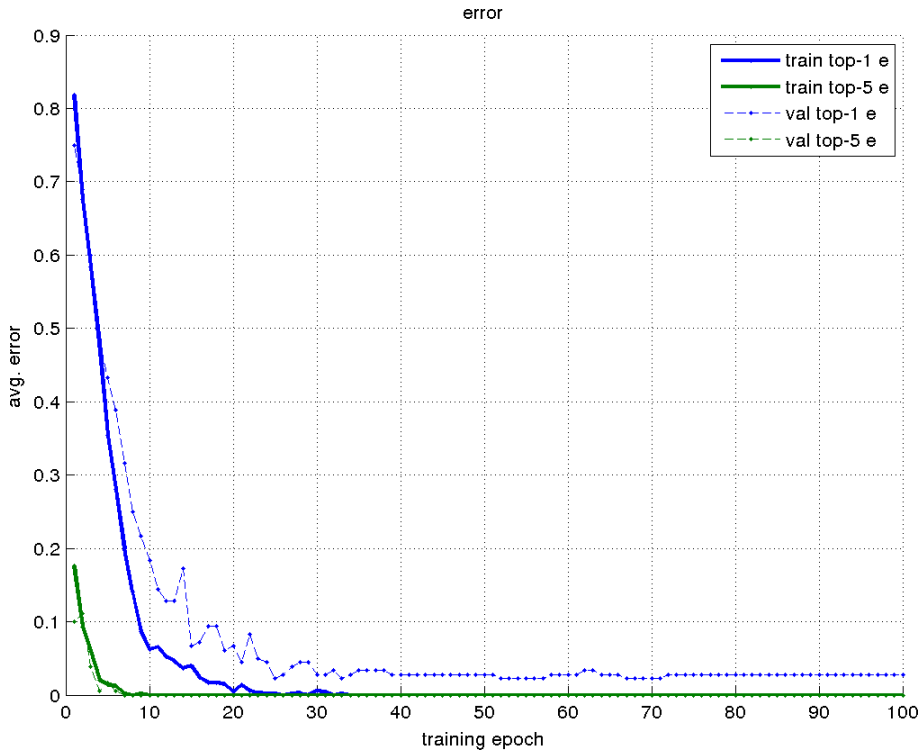
شکل۳: نتیجه‌ی اجرای ۵/۳، دقت بروی داده‌های تست: ٪۹۷.۲ انرژی شبکه بروی داده‌های تست: ۰.۰۶



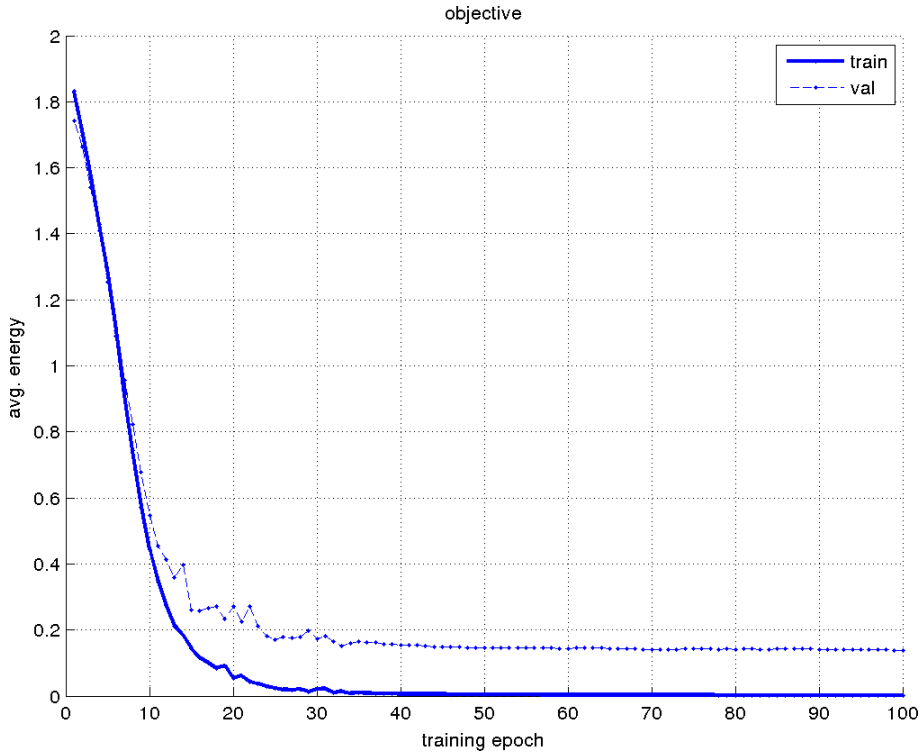
شکل۴: نتیجه‌ی اجرای ۵/۴، دقت بروی داده‌های تست: ٪۹۷.۲ انرژی شبکه بروی داده‌های تست: ۰.۰۸



شکل۵: نتیجه‌ی اجرای ۵/۵، دقت بروی داده‌های تست:۱۰۰٪ انرژی شبکه بروی داده‌های تست: ۰.۰۶



شکل۶: میانگین خطا ۵ اجرا به ازای ۱۰۰ اپوک



شکل۷: میانگین انرژی ۵ اجرا به ازای ۱۰۰ اپوک

```
Average train error:
        top-1: 0.000000
        top-5: 0.000000
Average validation error:
        top-1: 0.027778
        top-5: 0.000000
Average objective:
        train: 0.001982
        validation: 0.138697
```

شکل۸: جمع‌بندی کلی از نتایج اجراها