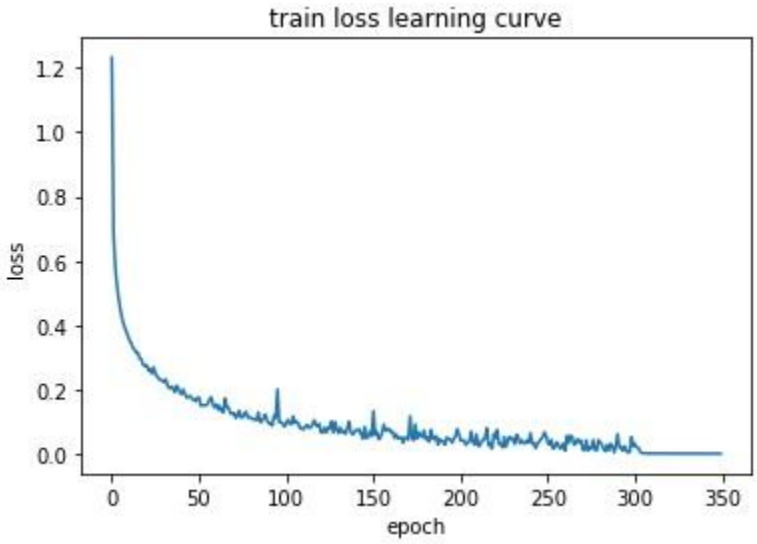
Convolutional Neural Network

თავიდან დავწერე სტანდარტული ნეირონული ქსელი, 3 hidden layer-ით, sgd ოპტიმაიზერით ნესტეროვის მომენტით, რადგან წავიკითხე, რომ ხშირად shallow ნეირონული ქსელებისთვის sgd-ს იყენებენ, ხოლო ნესტეროვის მომენტი ეხმარება რომ უფრო სწრაფად წავიდეს გლობალური მინიმუმისკენ და ლოკალურ მინიმუმში არ აღმოჩნდეს (ისიც ვნახე, რომ adam/adagrad sgd-ს ოპტიმიზირებული ვერსიებია, თუმცა ჩემს ქსელზე რომ გავუშვი NaN-ს წერდა loss-ზე რაღაც რაოდენობა იტერაციების შემდეგ) და loss=categorical_crossentropy-თ, რადგან, როგორც ლექციებში ვისწავლეთ, კლასიფიკაციის პრობლემებში ძირითადად ამას იყენებენ.

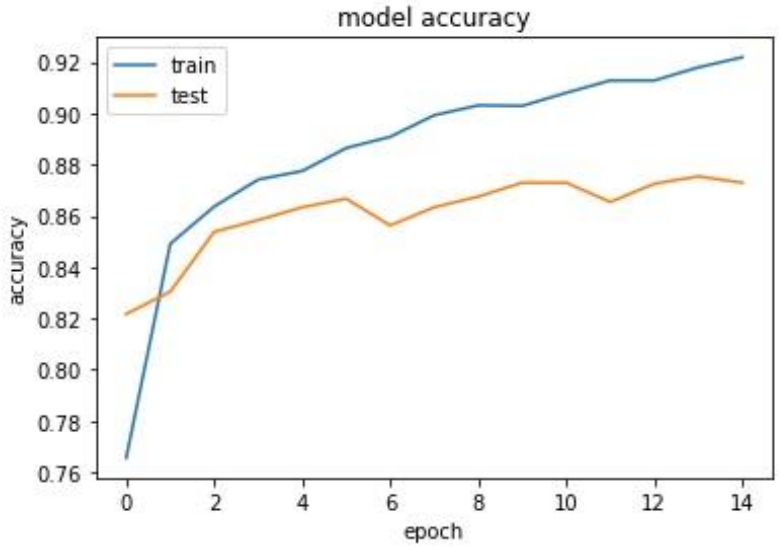


საშუალოდ 86-87% ჰქონდა ასეთ ქსელს accuracy cross-validation და test სეტებზე, ხოლო learning curve მარჯვნივაა მოცემული. მთლიანი training set-ის სწავლისას 300-350 იტერაცია სჭირდება convergence-სთვის.

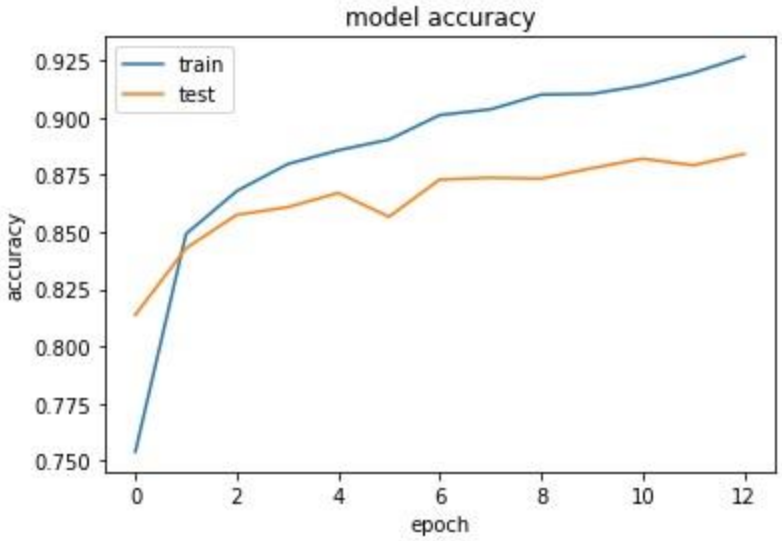
შემდეგ გადავწყვიტე კონვოლუციური ნეირონული ქსელიც მეცადა, რადგან ის ძალიან კარგ შედეგებს იძლევა Image classification პრობლემებში. შევქმენი 2 კონვოლუციური შრე, 12 და 10 ფილტრით შესაბამისად, მეტჯერ გაკეთებაც ვცადე კონვოლუციის და მეტი ფილტრებითაც, თუმცა შედეგი თითქმის იგივე იყო, ზოგჯერ კი უარესიც, ამიტომ ვარჩიე ნაკლები დამეტოვებინა, რათა ნაკლები დრო დაეხარჯა სწავლაზე. ფანჯარა ორივეგან 3x3 ავიღე, კონვოლუციის ჩასატარებლად, რადგან, არც იმდენად ბევრია, რომ ვერ ისწავლოს კარგად ამის გამო და არც იმდენად ცოტა, რომ overfit გამოვიდეს. აქტივაციის ფუნქცია პირველში სიგმოიდი მაქვს, ხოლო მეორეში რელუ, აქ ბევრნაირი ვარიანტი ვცადე, თავიდან სტანდარტულად ორივეგან რელუ მქონდა, თუმცა, ასე უკეთეს შედეგებს იღებდა და ეს ვამჯობინე. ორი კონვოლუციური შრის შემდეგ დავამატე AveragePooling შრე 2x2 ფანჯრით, feature-ების და, შესაბამისად, გამოთვლის დროის შესამცირებლად და ასევე overfitting-სგან თავის დასაცავად. უფრო ხშირად MaxPooling-ს იყენებენ, თუმცა, შედეგი თითქმის ერთი და იგივე ჰქონდა ამ 2-ს და AveragePooling უფრო ლოგიკურად მომეჩვენა, რადგან greyscale-თაა სურათები. ამის შემდეგ დავამატე 2 Dense layer, ერთი

მიღებული მატრიცის დასამუშავებლად, რასაც დიდი რესურსი სჭირდება, ამიტომ 100 ნეირონი ავიღე, ხოლო მეორე output-სთვის 10 ნეირონიანი, რადგან 10 კლასი გვაქვს სულ. ბოლოს წინა შრის აქტივაციის ფუნქციად სტანდარტულად რელუ ავიღე, ხოლო ბოლოსად softmax, რადგან წავიკითხე, რომ ძირითადად ბოლო შრეზე ან sigmoid-ს იღებენ, ან softmax-ს, ორივე ვცადე და softmax-ს უკეთესი შედეგი ჰქონდა. Weight-ების ინიციალიზაციისთვის გამოვიყენე ხავიერის თანაბარი ინიციალიზაცია, რადგან ის აგვარებს ზედმეტად პატარა ან ზედმეტად დიდი წონების პრობლემას (ორივე გამოუსადეგარს ხდის მონაცემებს, რადგან ასეთი წონებით ვერ სწავლობს ქსელი); ეს ინიციალიზაცია ადგენს თითოეული წონის range-ს input/output ნეირონების რაოდენობის მიხედვით. Pooling-ის გარდა, overfitting-ის თავიდან ასარიდებლად ვიყენებ წონების რეგულარიზაციას, რომელიც loss-ს უმატებს რაღაც პარამეტრი * წონების კვადრატების ჯამზე, რაც უფრო პატარაა ეს პარამეტრი, მით მეტად აძლევს overfitting-ის „უფლებას“ ქსელს, რადგან წონებს უფრო დიდ range-ს აძლევს, ხოლო, რაც უფრო დიდია, მით უფრო პატარა უნდა იყოს წონები, ამიტომ პარამეტრის ზრდასთან ერთად, მოდელი underfitting-სკენ მიდის. ეს პარამეტრი გადავარჩიე 0.1-დან 0.001-მდე, 0.1-ზე underfitting იყო, 0.001-ზე ჰქონდეს ყველაზე კარგი შედეგი და overfit-ც აშკარად არ იყო, ამიტომ ეს დავტოვე. Loss ფუნქციად ისევ categorical_crossentropy ავიღე, ხოლო ოპტიმაიზერად adagrad, რადგან, ამჯერად, როგორც მოსალოდნელი იყო, მას გაცილებით უკეთესი შედეგი ჰქონდა, ვიდრე sgd-ს.

თავიდან 15 იტერაციაზე ვუშვებდი, თუმცა, როგორც მარჯვნივ გრაფიკზე ჩანს, დაახლოებით 13-ის მერე cross-validation-ის accuracy იკლებს, ხოლო train-ის იზრდება, რაც overfit-ზე მიანიშნებს, ამიტომ შევამცირე იტერაციების რაოდენობა.



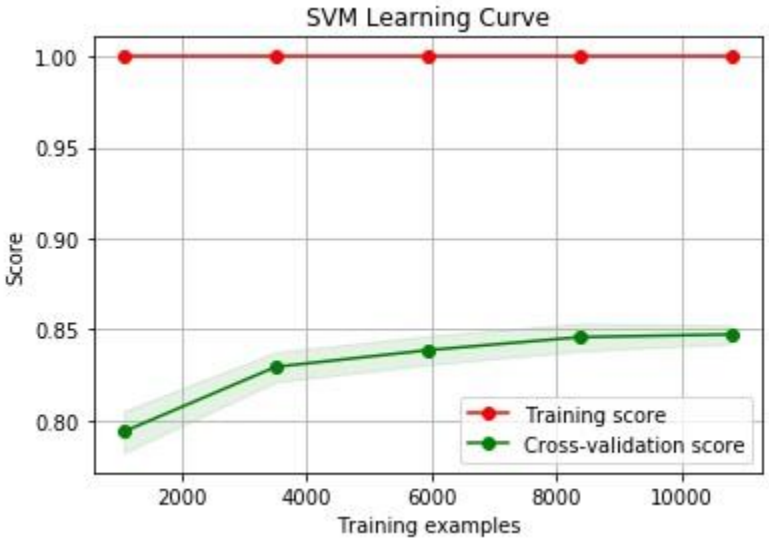
საშუალოდ ამ ქსელს 88-89% accuracy აქვს test და cross-validation სეტებზე. მე-13 იტერაცია არ ჩანს გრაფიკზე, თუმცა, 88-89%-მდე ადის ხოლმე ძირითადად.



ჯერ ყოველთვის cross-validation set-ზე ვამოწმებდი მოდელს, რომელსაც training set-ის რანდომ 20%-ად ვირჩევდი, რადგან არ გამოსულიყო რომ მოდელს მხოლოდ ფიქსირებულ test set-ზე ვარგებ და ახალ ტესტზე აღარ ემუშავა კარგად.

Support Vector Machine

SVM-ში ყველა უკვე არსებული kernel ვცადე, poly-ზე იყო საუკეთესო შედეგი, ყველაზე კარგად აკლასიფიცირებდა, თუმცა, default-ად ხარისხი 3 აქვს და 2-მდე შევუმცირე, რადგან overfit-ის ნაკლები შანსი ყოფილიყო, ასევე tolerance for stopping criteria გადავარჩიე და საუკეთესო შედეგი 0.4-ზე მქონდა, თავიდან იყო 0.01 და, სავარაუდოდ, Overfit ხდებოდა რადგან უფრო დიდხანს ისწავლიდა. აქაც ჯერ training set-დან რანდომად ამორჩეულ cross-validation set-ზე ვტესტავდი და მხოლოდ ამის შემდეგ მოცემულ ტესტზე, რომ ფიქსირებულ ტესტ სეტზე არ ყოფილიყო მორგებული. საბოლოოდ accuracy 85-86% გამოვიდა cross-validation-ზე test-ზე.



ესეც learning curve svm-სთვის. Training set-ზე, ცხადია, ყოველთვის ერთია accuracy, ხოლო cross-validation-ზე 85-სკენ მიდის როგორც ვხედავთ, cross-validation ის accuracy აღარ იზრდება, ანუ საკმარისად ისწავლა უკვე.