Yelnur Shauketbek

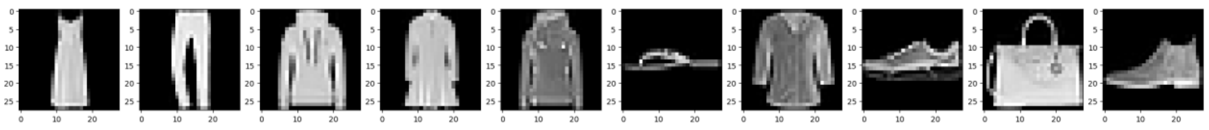
Student ID: 2078709

January 26, 2023

Cognition and Computation: Individual project

1. Introduction

In this project I chose the same Python code, training and testing datasets are FashionMnist. Training dataset consist of 60000 samples, test dataset contains 10000 samples. Our samples are 28x28 pixels items and pixels have values between 0 and 1, 0 is a black pixels, 1 is a white pixels. Some samples from this dataset:



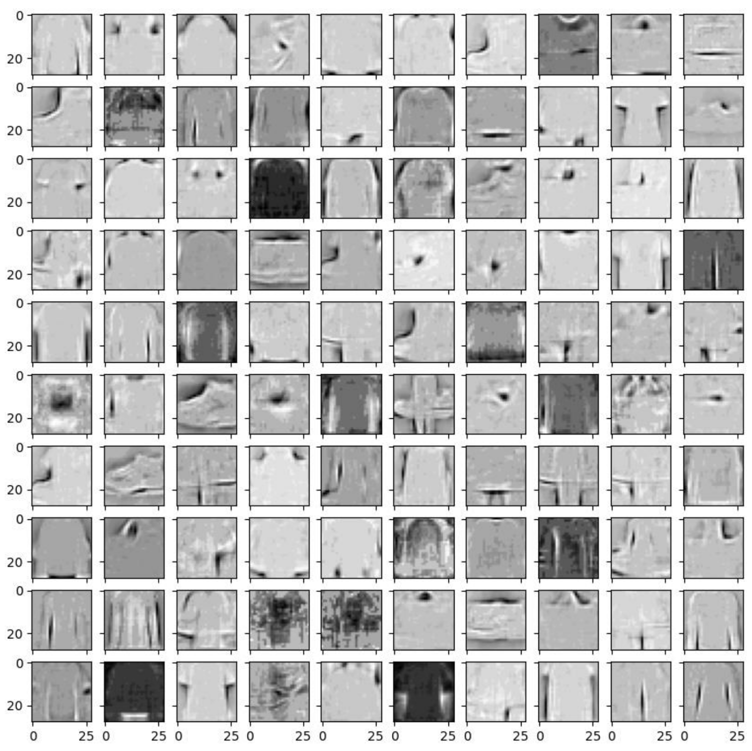
I have used pytorch with cuda, because is more faster to use gpu than cpu.

2. Models

2.1 First Model

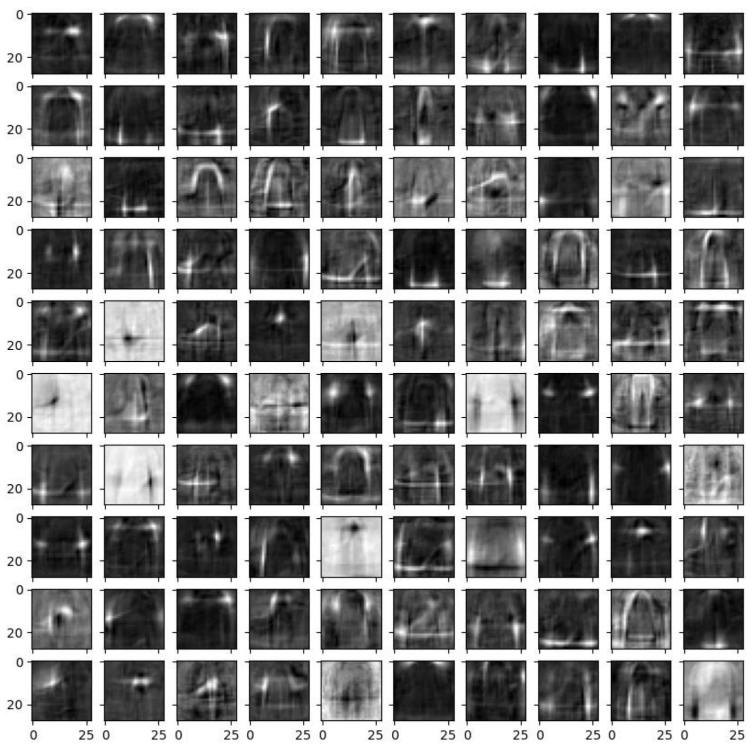
First I start with DBN (Deep Belief Network), it is unsupervised deep learning architecture, which consist of Restricted Boltzman Machines. Hidden_units or hidden neurons in layers are [400,500,800]. It's standard settings we used in Lab practices. Then we train this network using 50 epochs per each layer. Therefore we have trained 150 epochs for network. We expect, first layer is the worst and we expect that the third one become the best one. It's true because for the first layer we have average recursion error around 1.6918 and standard deviation error 0.0762, but for the third layer we have: 0.7002 and 0.0233 for average and standard deviation errors as well.

Let's see our learned weights. This is our first layer's weights. We expect to see some shapes that our network learned from training.



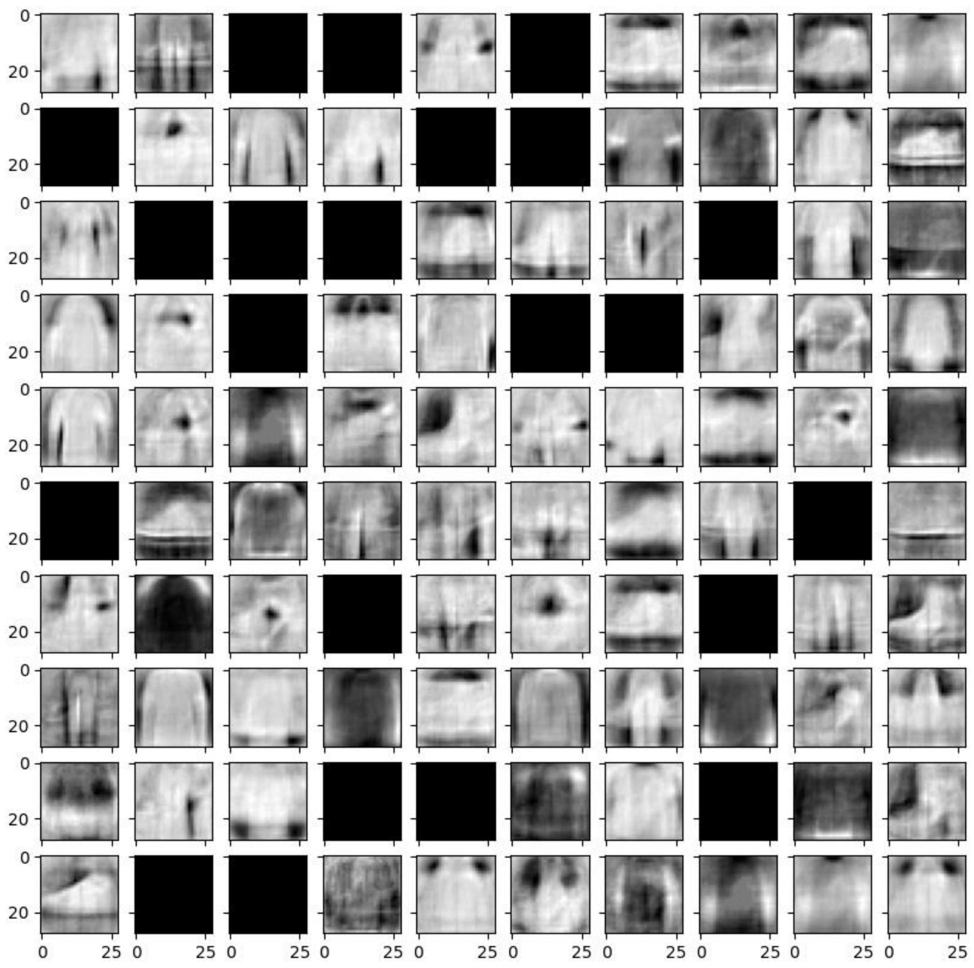
Pic.1 First layer's learned weights

It seems like figures and shapes of the items and all of them are almost invisible, because of the background, I mean background and item shapes are the same gray, but it's the only first layer. Let's look at the second one.



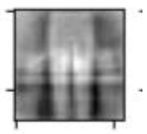
Pic. 2 Second layer's learned weights

Now we have another situation, almost all backgrounds are black, but we can see that now shapes are mostly black, but somewhere we clearly see gray or white silhouettes. Let’s look at the last third layer.



Pic. 3 Third layer’s learned weights

Third layer’s weights look better than previous ones. Here we almost see shapes and silhouettes of the items, for example this one:

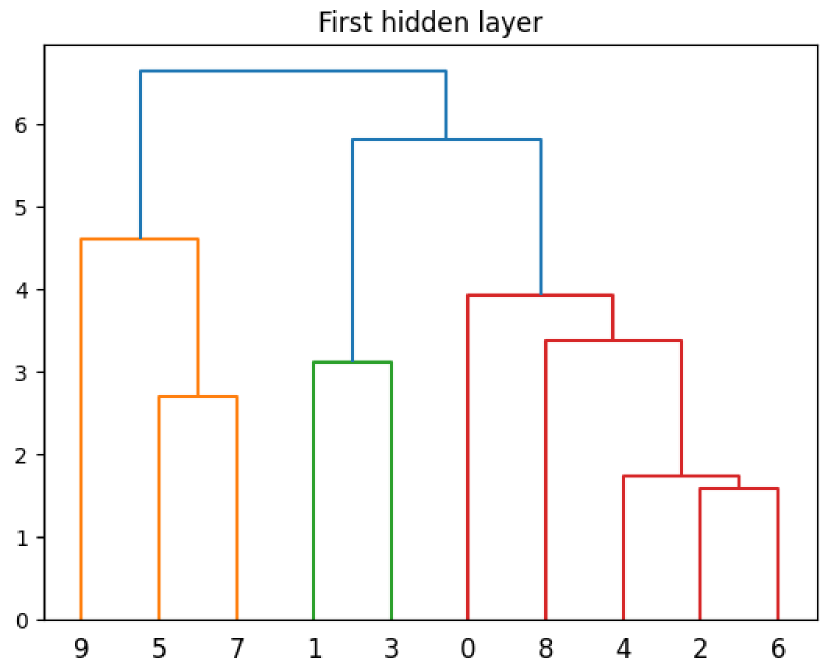


It seems like pants or trousers and I’m sure this is the trouser in our trained model. Somewhere it’s still hard to find out what item is it in the Picture 3, but it’s possible.

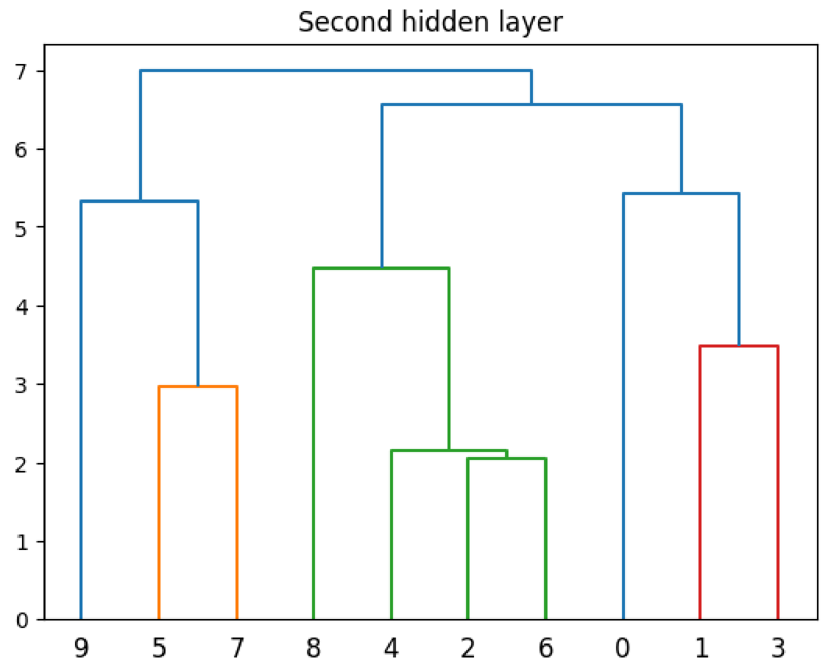
Now let’s look at the dendrograms of layers.

Our labels: 0 – T-shirt/top; 1 – Trouser/Pants; 2 – Pullover; 3 – Dress; 4 – Coat; 5 – Sandal; 6 – Shirt; 7 – Sneaker; 8 – Bag; 9 – Ankle boot.

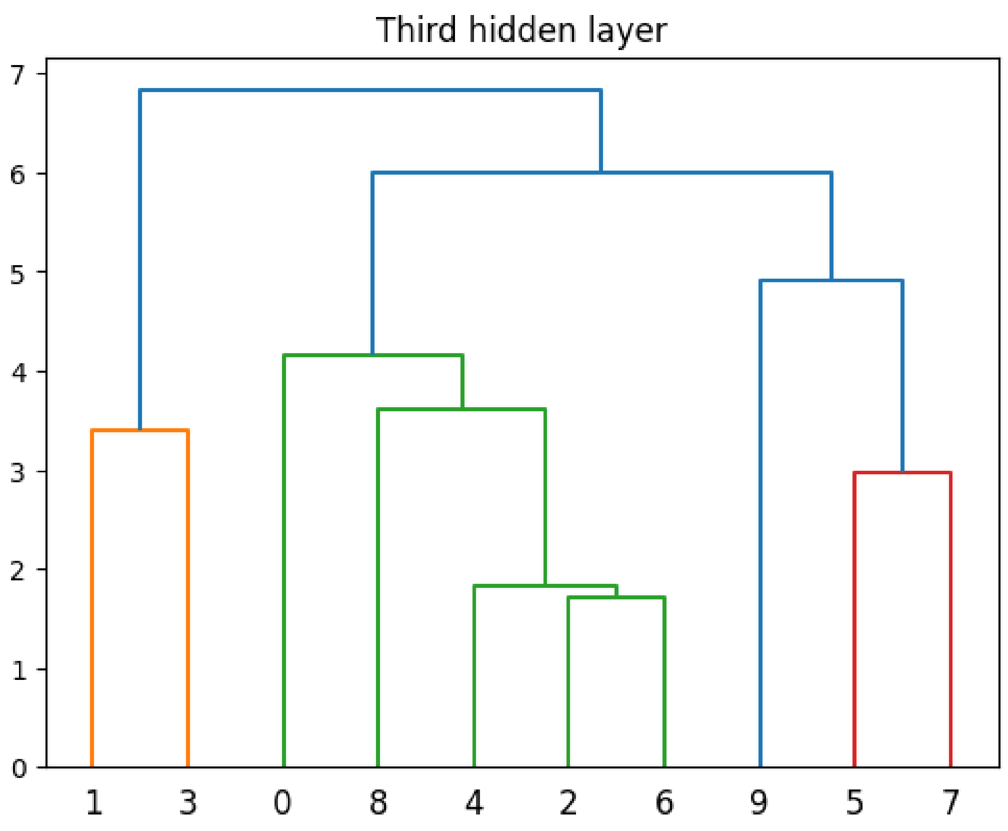
We expect that the third layer will be different than first and second one, because we know that from layers to layers connections of neurons could change.



For the first layer we have that 0, 8, 4, 2 and 6; 9, 5 and 7; 1 and 3 are very similar for our model.



For the second layer we have that 8, 4, 2 and 6; 1 and 3; 5 and 7 are very similar for our model. Let’s look at the third dendrogram.



For the third layer we have that 1 and 3; 4, 2 and 6; 5 and 7; are very similar to our model. Actually 4 (coat), 2 (pullover) and 6 (shirt) really similar to each other. It's even hard to human to say which of these samples is which. So thanks to dendrograms we could see how our model classify items.

Now we can readout representations of each layer. For that we can use linear model. Then we need to train linear model and we use 1500 epochs and we expect that for each next layer loss will decrease. For the first layer we have 0.4611 loss, second: 0.4414, third: 0.4458. Now we should calculate accuracies for each layer. And we have: first layer – 0.83, second layer – 0.84, third layer – 0.84. That mean 84% for the third layer.

2.2 Second Model

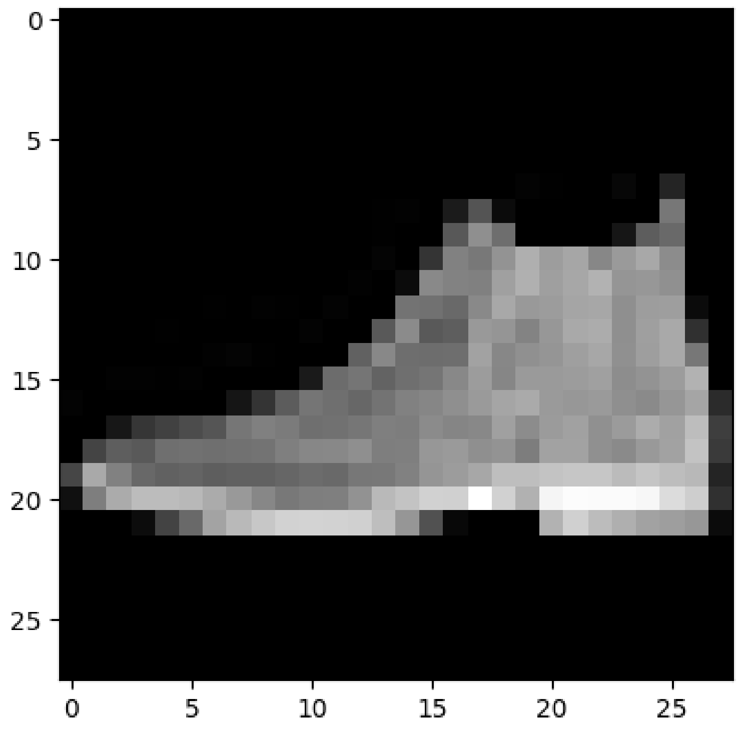
Second model that we use is Feed-Forward neural network model or FFNN model. FFNN is an neural network where connections are straight. We need it to compare two models. As before it will have 3 hidden layers and [400,500,800] neurons per each layer. We need to train it, 1500 epochs. We expect decreasing of loss, and it truly decreased. After 1500 epochs we have loss around 0.4438. Let's compute the accuracy of this model and it gives 0.8335 or 83.35%. A little bit worse that DBN model.

3. Adding noises

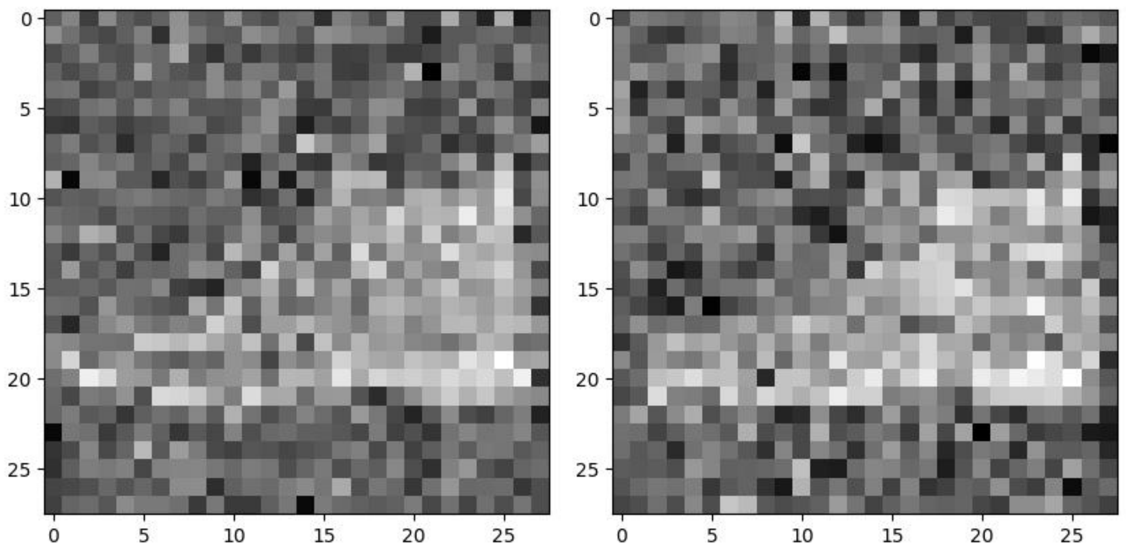
3.1 Random noises

Now try to add some random noises in the sample.

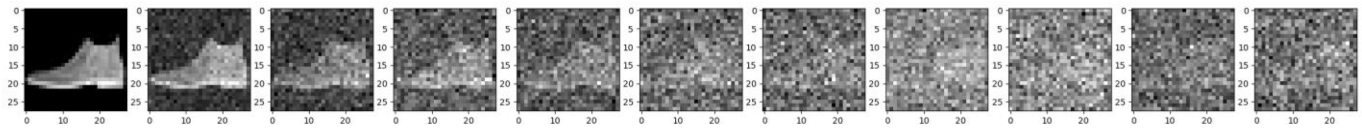
For example, how it looks without any noise.



We perfectly see ankle boot. But how it looks with added random noises. How we can prove that noises are random? It’s easy, we need to call inject_noise function twice, and we will have two different samples with noises. In the left side we can surely say that this picture has more gray pixels, on the other hand right side picture has more black pixels.



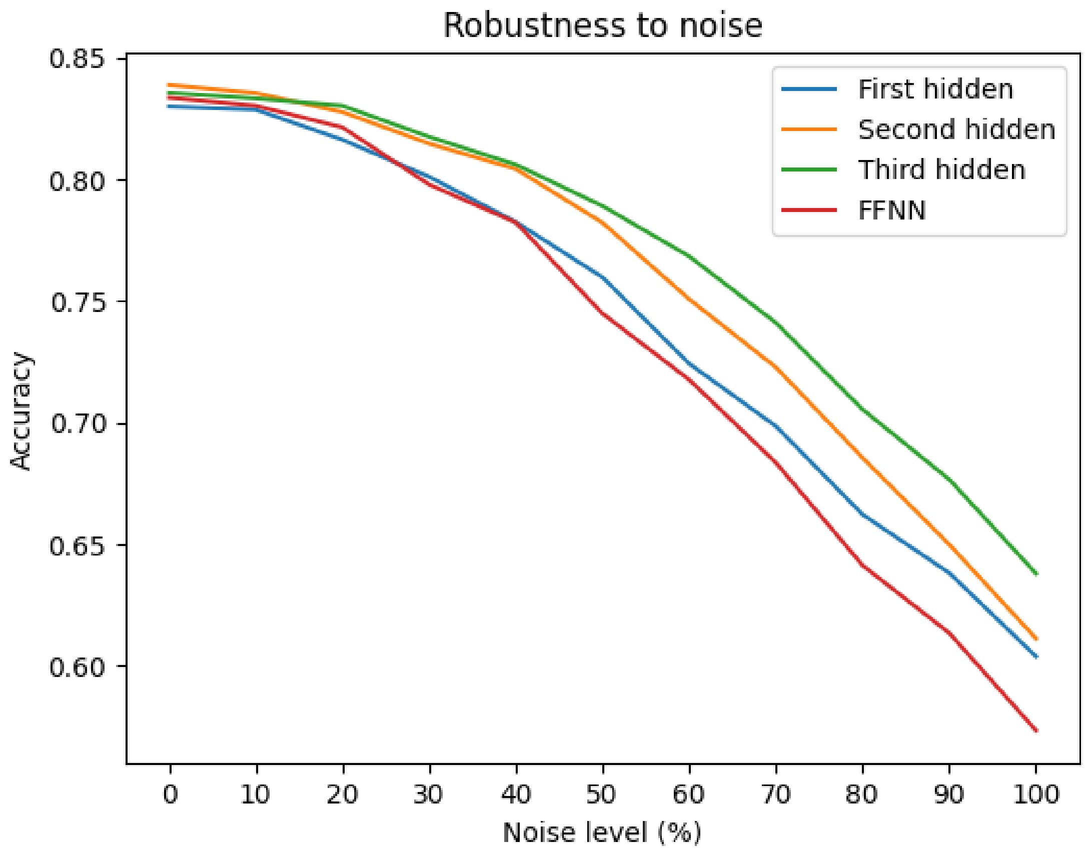
And of course we can change noise level.



From the left to right noise level increase from 0 to 1, by step 0.1. As we can see before 0.5 noise level we could understand that it is an ankle boot, but after 0.5 for human there are almost only noise without any items in it. And now let's see how DBN and FFNN models will deal with these noises.

Noise level	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
DBN	0.836	0.835	0.829	0.820	0.806	0.788	0.761	0.746	0.705	0.669	0.644
FFNN	0.834	0.831	0.819	0.802	0.780	0.751	0.716	0.683	0.643	0.605	0.579

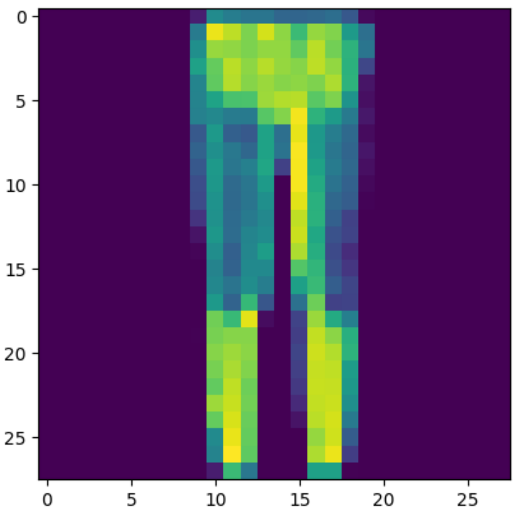
As we can see from this table Feed-Forward network almost have the same result as Linear model for 0.1 and 0.2 noises, and from 0.3 FFNN accuracy droping down. These results clearly show us that DBN model doing well with random noises, for example when noise level is equal to 1, Accuracy of H3 read-out: 0.644. It's a great result! With full noise our model shows almost 65% accuracy. On the other hand our FF network shows 58% accuracy on noise level = 1. And we will clearly see it in graphic below.



3.2 Special noises (adversarial attacks)

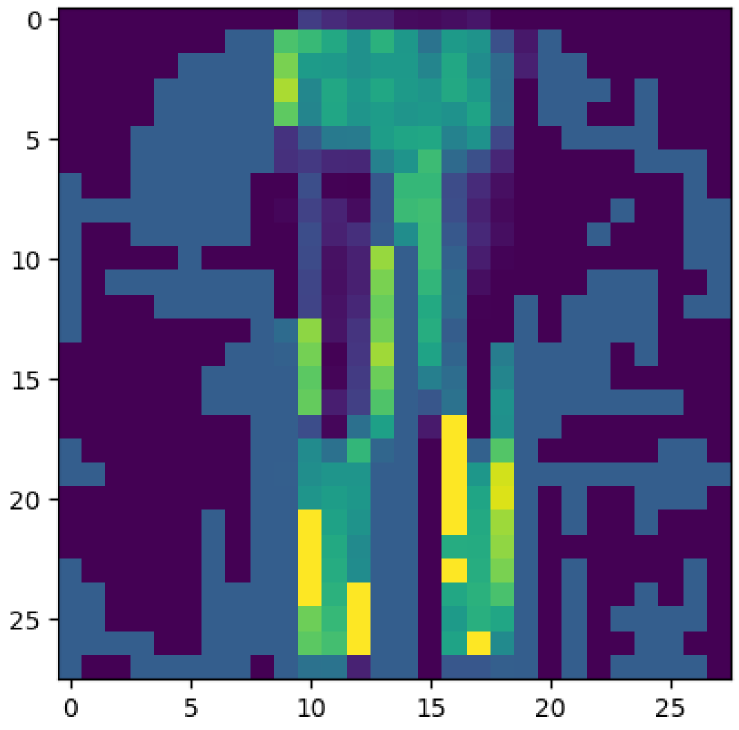
Now let's add special noises, which called 'adversarial attacks'. Fast gradient sign method is the method where the noises adding exactly in a special pixels, that makes models to classify incorrectly.

Let’s look how sample looks without adversarial attack.



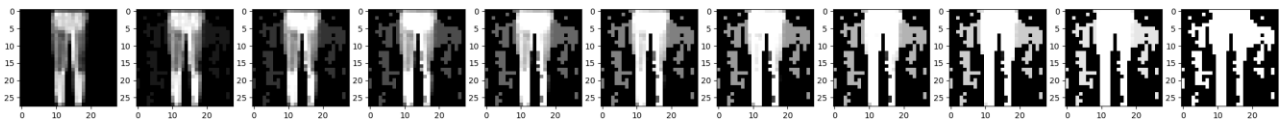
For this sample FFNN and DBN with readout make a right prediction and decision, they chose 1 category, which is trousers and pants.

Now time to attack the sample and try to confuse two models. Let’s look how sample looks like after attack.



And we are not sure what type of items is it. Attack was good and models made a wrong predictions. FFNN predicted that this attacked sample is 3 (dress). DBN with readout predicted that this attacked sample is 8 (bag).

Let’s see how changing our sample from the left epsilon = 0 to the right epsilon = 1 by step 0.1 epsilon.

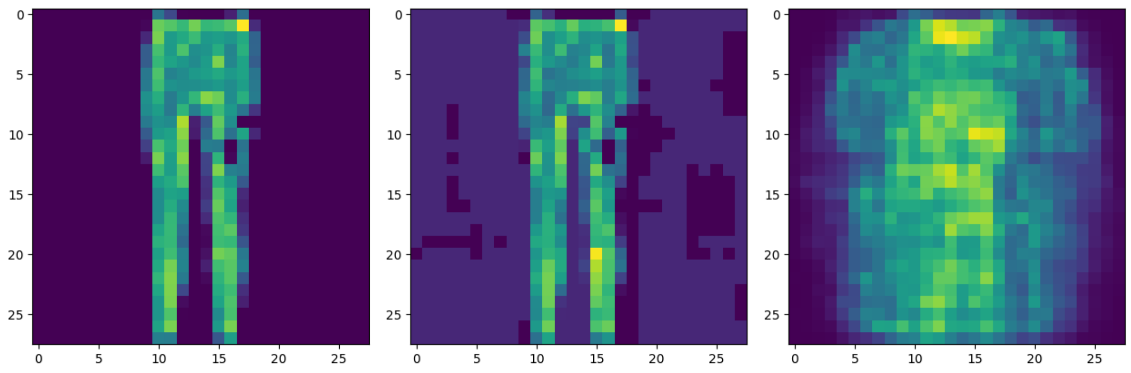


And let’s see accuracies.

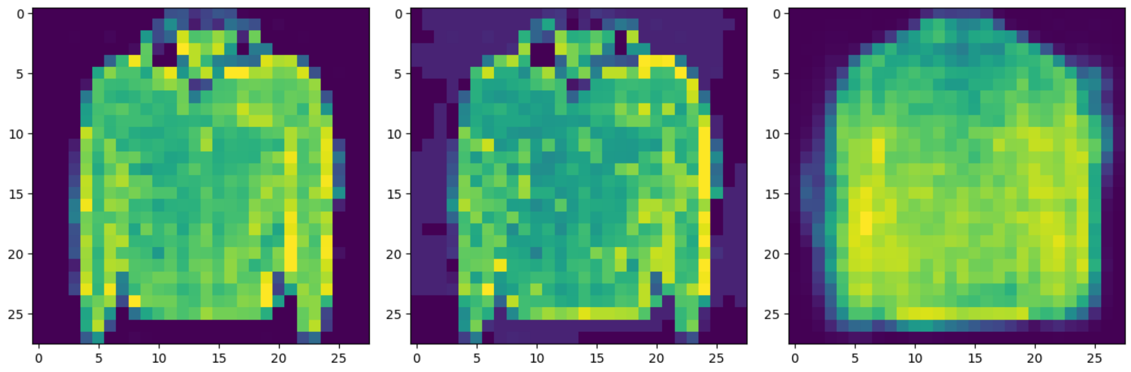
Noise level	Epsilon = 0.1	Epsilon = 0.2
FFNN	30.42 %	2.55 %
DBN with readout	40.07 %	12.73 %

Now we need to try top-down reconstruction on DBN with readout, with 1 and 2 steps in it.

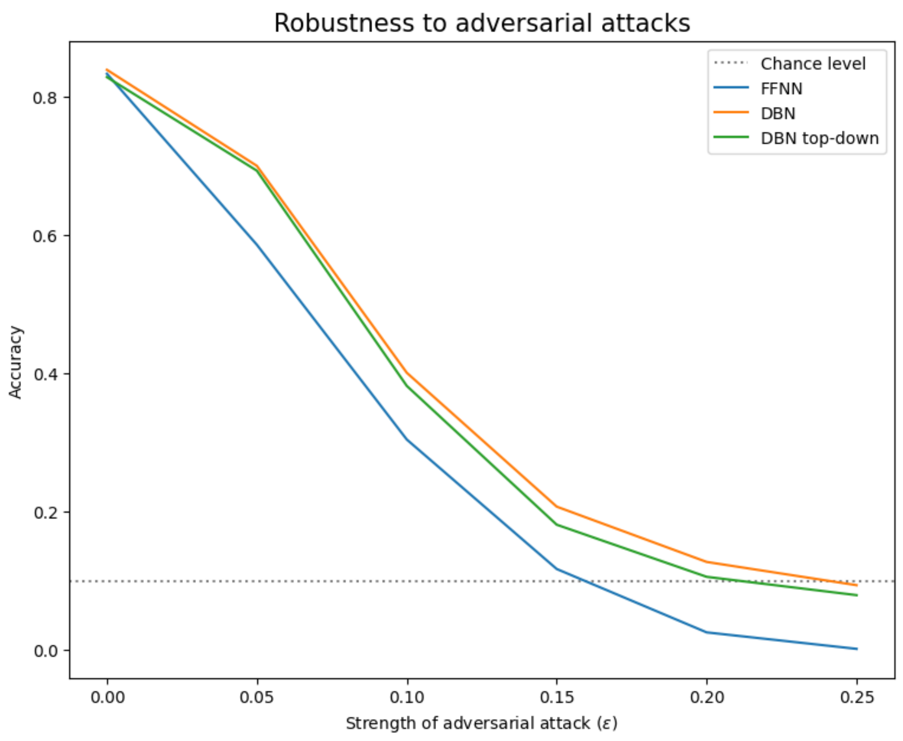
First with 1 step top-down reconstruction. Left picture: original sample. In the middle: perturbed or attacked sample. Right picture: 1-step reconstructed sample. Looks not so good, but accuracy increased to 38.17 %.



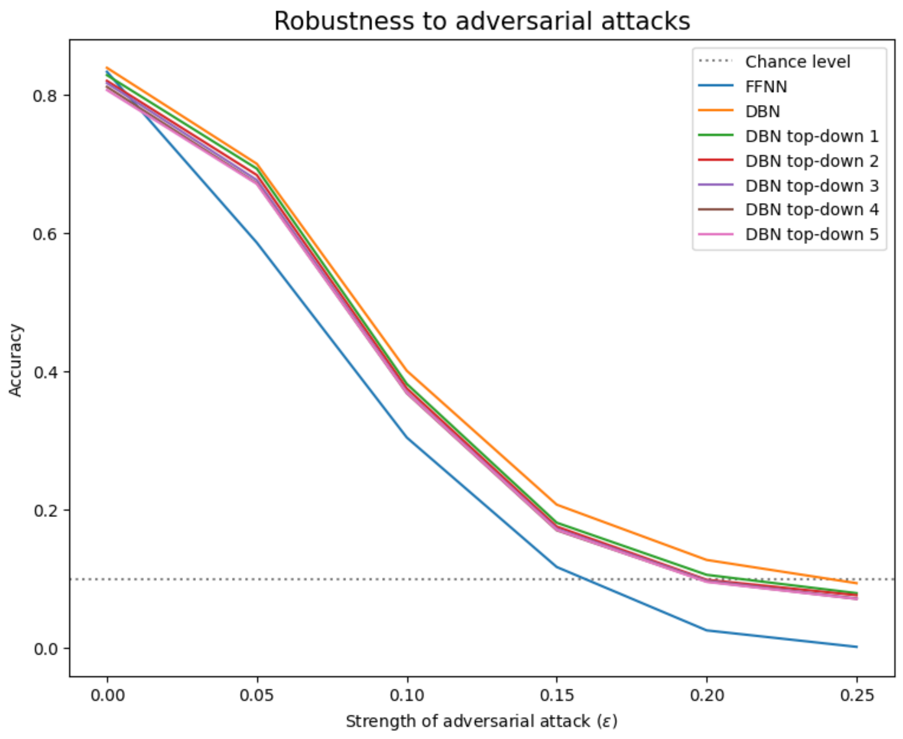
Now with 2 steps. As before: Left picture: original sample. In the middle: perturbed or attacked sample. Right picture: 1-step reconstructed sample. Accuracy increased, but only 37.52 %, that a little bit worse than 1-step reconstruction.



Now I want to check accuracies when epsilon is changing from 0 to 0.25. In this plot we have FFNN, DBN and DBN with 1 step top-down reconstruction. We expect that graphic will go down with epsilon increasing. Let’s check it out.



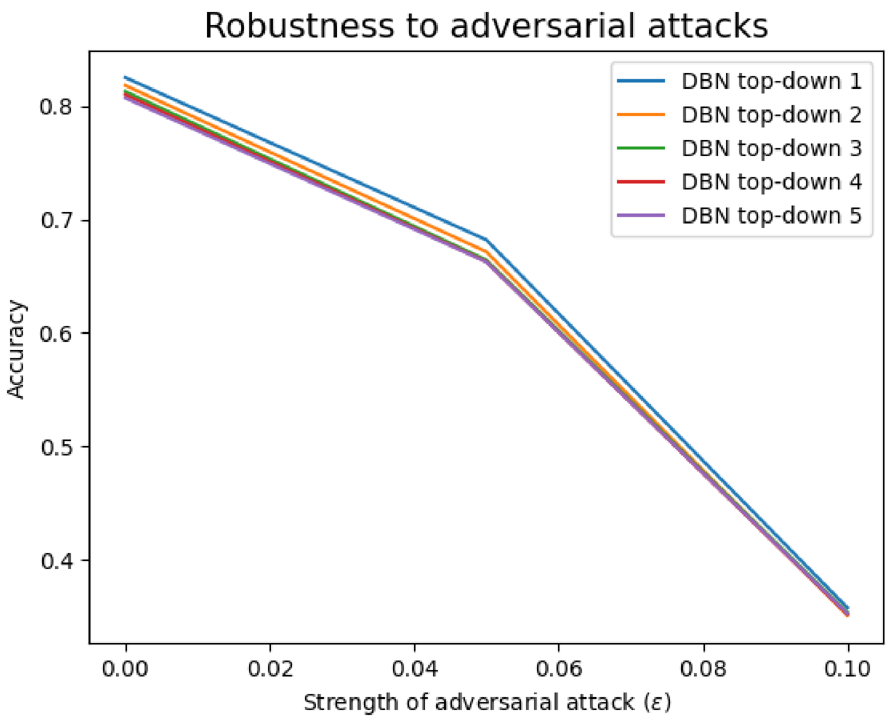
As we see from the plot, common DBN with readout makes better predictions than DBN with 1-step top-down. Is this true for all DBN with n-steps top-down? I want to check from 1 to 5.



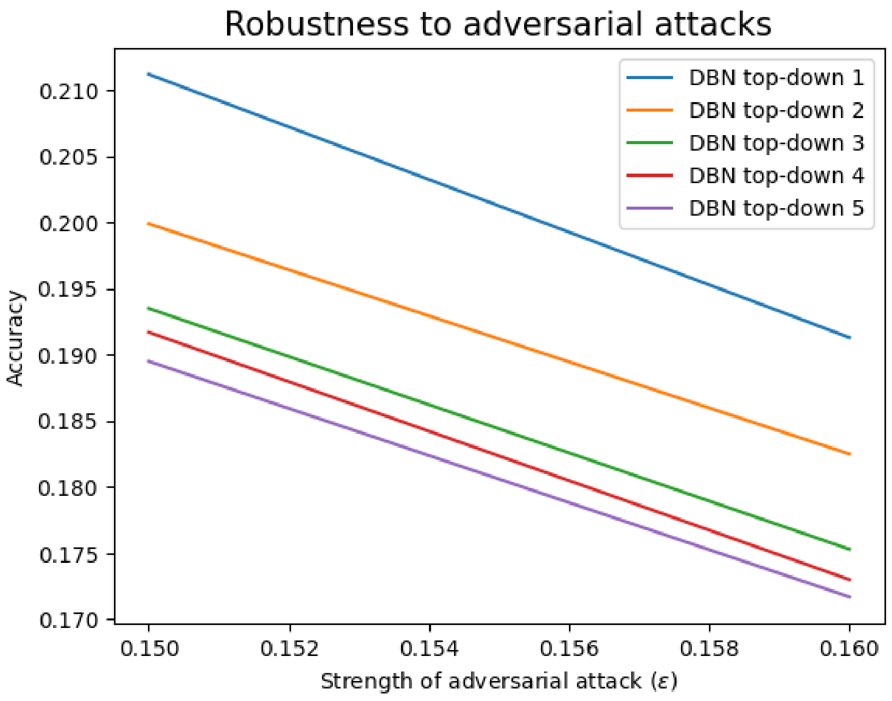
As we see in this new plot, common DBN is really better than DBN with n-steps top-down reconstruction.

Now I want to check which one of 1,2,3,4 and 5 steps are the best. Because in the graph above we can't understand it and seems like they are all in one line.

Let's look closer to short area from 0 epsilon to 0.10 epsilon.

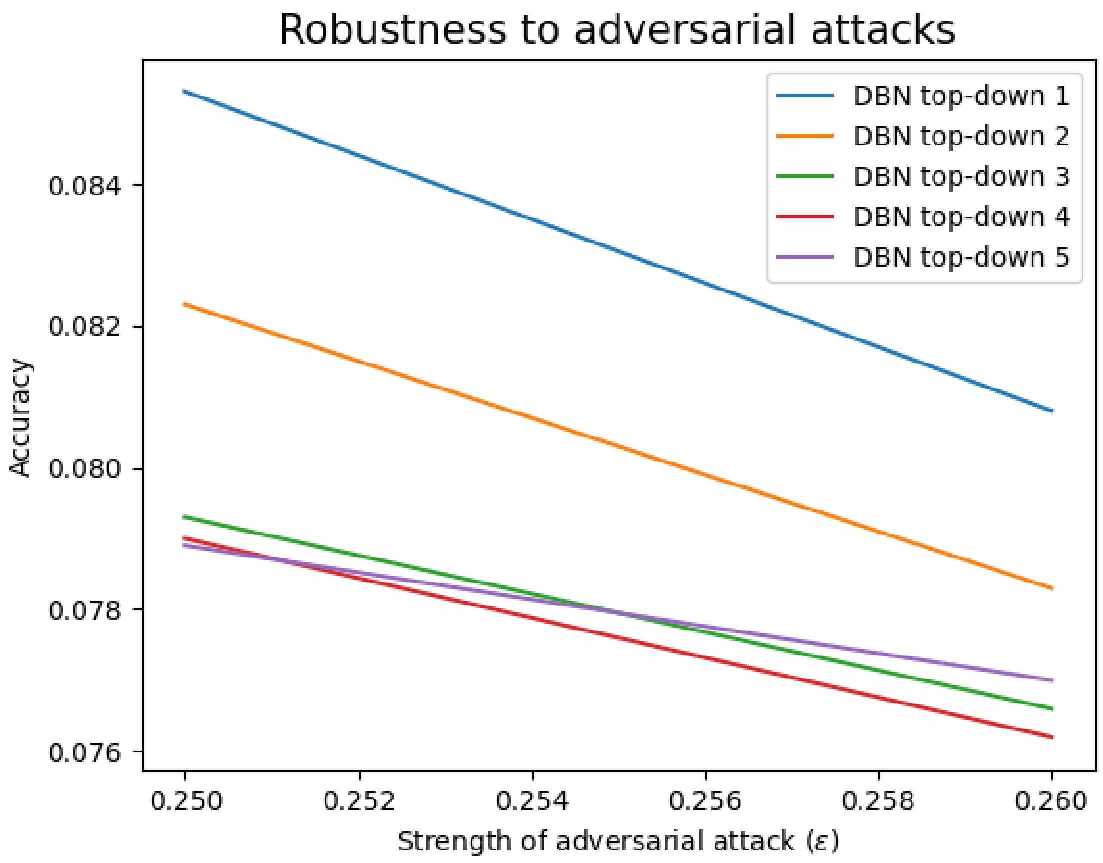


Let's look closer to short area from 0.15 epsilon to 0.16 epsilon.



Thanks to these two graphs we can understand that DBN with 1-step top-down is the best from 0 to 0.16.

And we need to check the last area from 0.25 to 0.26 epsilon.



We can see that 5-steps becomes better than 3 and 4, but 1-step top-down is still the best. So there is no need to use 2,3,4 and 5 steps, because 1-step is better than them. But as I wrote before just common or standard DBN with readout better that DBN with 1-step top-down. So for this Fashion dataset using top-down reconstruction is meaningless.

4. Conclusions

1) DBN and FFNN get a good accuracy on samples without any noises. More than 80%.
2) When we add random noises to samples, we get a little bit worse results, when noise level not so big. And when we get full noise level, i.e. 100%, we get accuracies: for DBN around 65% and for FFNN around 58%.

3) When we add special noises or adversarial attack samples, we get a huge difference between random noises results. For only 0.1 noise level, we have 30% accuracy for FFNN and 40% for DBN. But if we increase noise level to 0.2, we have a really huge difference for models, therefore, around 3% for FFNN and 13% for DBN. That's really huge accuracy dropping.
4) And if we use DBN with n-steps top-down reconstruction, it's not making any sense, because the results become worse than standard DBN with readout. But I need to mention that using DBN with n-steps top-down reconstruction is meaningless only for this Fashion dataset, in other datasets it could give a great accuracy increasing.
5) And one of the biggest causes of dropping accuracies is similarity of samples from different classes. As I mentioned before classes 4 (coat), 2 (pullover) and 6 (shirt) really similar to each other. Even I can't understand from the first time where pullover and where coat. There are really similar. Above picture that contains every sample which are represented in this Fashion dataset. And, of course, we need to understand that every new training gives us new data, so there are some kind of randomness influence to our results.