Modern Computer Vision

Prof. A.N. Rajagopalan

Department of Electrical Engineering

IIT Madras

Lecture-24

Okay, so I will just write down the expression, okay, yesterday the box convolution that we had, I think we all understood what it is but let me just write it down. So it is like yij is equal to summation m equals 0 to k1 minus 1, n equal to 0 to k2 minus 1, then sum over all channels c equal to let us say 1 to capital C, then i of i minus mj minus n the channel into h of m, n that channel, right. In general, okay, in general wherever you are, right, when you are doing a box convolution, right, this is how it is. So if you have multiple feature maps, then it will go, so it is like this, right, if at the input if you had 3 channels, then the c will go from 1 to 3 and i is that particular image right on which you are acting or a feature map. If it is somewhere in between the, in between in that network, then it will be one of the in between channels. If it is at the input, then it will be i itself.

So it can be like a color image, you know that has 3 channels, r, g, v and b, there is a color, red color, green and blue. So it could be those channels and then you have a filter which is for, right, which is actually for that channel and therefore, when you combine concatenate for all c's, you will get actually a box filter and you are just summing up, right, like I said, I mean, what you have is this kind of, you know, so you have, you have this kind of a box filter, right, that is actually summing up, summing up along, along the, I mean, a channel depth, right, if you want to think about the channel depth. So you have got multiple channels and you are applying a box filter and that box filter is simply calculating a convolution over each feature map and all of them are being summed up, right and k1 and k2 are like the size of your filter, right, h, the size of h is actually k1 by k2, right. So depending upon the, it can be symmetric, it need not be symmetric, right, so in general it will be like some k1 by k2.

You can think of situations where, you know, h need not be square, okay, so it is not going to say necessary that h should be square and so you are summing up over k1 by k2 which is the dimension of h and you are summing up over all, all channels, right, c equal to 1 to, 1 to the capital C. So capital C will be 3 if you are at the input and if you have a color image, capital C will be 1, I mean, if you are at, if you just have 1 channel right at the input, okay. Now having said this, let me actually go to, go to an animation, right, which I, which I thought I will show to you, right and so I think this you saw us today and of course, you know, right, this is what is actually is a document kind of a reader, right. So you can think of, you know, a digit 0 to 9, okay and what is appearing, right, is here, what is being, you know, input and it is supposed to, you have a network that is supposed to tell what is the, what is the right answer, right, what kind of a digit is this and the inputs as you can see, right, the way they are, they are not very structured, right, they are like the way you write them, like you do not really write in a very structured way. But irrespective of, so these training samples if you see on the left, right, so the kind of training and all that you give is like, you know, you will give all kinds of things and then it is always better to expose a network to various variations of the input.

In this case it is just a digit and therefore, right, you want to kind of expose it to various variations that can occur and the network is supposed to sort of figure out, right and after training if you input a digit it should suppose, you know, it should tell what that is a digit is. So these are all very small images by the way, right, this is like 1989, 60,000 training samples, 10,000 test samples, output is an integer between 0 and 9 and you can see here that the way you train is that, you know, you give various translations of the, of a digit, rotations of a digit scale, right, I mean you may, some people may write big, some people may write small, then you can have squeeze and stroke width, right, people, some people may, you know, write bold, some may write, you know, less bold, noise, so you train against all of that. So like I keep saying, right, a lot depends on how well you train, I mean that is, that is kind of a bottom line, right, in terms of how rich a dataset you have, right, the more rich it is the better it is for a network because then it is exposed to different variations and if it has seen those variations then, or at least, you know, things that are close to that then there is every possibility that it will give out a proper output, okay. And yeah, this I think we have already done, so this, so this local, local connectivity and the parameter sharing but then before that, right, I mean there is something called a pooling layer, okay, which is something that we have not done yet, it could also have been done for MLP, okay, but we did not do it at that time and generally it is used for 2D, okay, what is called max pooling, it is generally called pooling. We will come to that in a minute but then before that I thought I will show you these animations, okay, which kind of tell you, okay, some of these are animations, some are not.

So to just tell you, right, what does local sort of a connectivity, right, what it means and what the parameter sharing actually means and you have already seen it yesterday, I wanted to show that first so that you have a picture in your head in terms of how those filters act and you know, the output channels, input channels and all right. When I say channel, I mean a feature map, okay, that is an accepted sort of, you know, terminology. If I say I have got n number of channels, what I mean is n number of feature maps and filters are typically the weights, okay, so filter means it is that kernel containing the weights, okay, that will be normally a terminology that we will, that we will continue to follow and here, right, if you see, if you have, you know, an MLP, right, then you see that if I have, so here is your, here is your input, okay, the blues are your input and then the reds are your, you know, next layer and you know that, right, if you had to learn the weights or if you had to train a network, I mean, you would have like, you know, if you just consider the weights, you have got like 7 into 3, right, which is actually 21, this is just for a small network. But suppose you said that, that I just assume local connectivity, okay, which means that, which means that this guy, right, instead of seeing all of them, all 7, just like we saw yesterday, right, you see, I took an image as an example, because in an image it is easier to sort of, to sort of understand this locality business, right, you can also do it for audio and all, but people are more accustomed to seeing images, you know, the interactions being local and so on, and which is exactly what is happening here. So you say that, so you say that, right, you basically every neuron fixates on a kind of a local region, right, so for example, so it will only fixate on these 3.

Yesterday what you saw, saw was actually a spatial grid, you saw a 3 cross 3 or a 5 cross 5 region. And I said, and as I said, a receptive field is something that can become bigger and bigger as you go right down the network, right, what a neuron ends up seeing, you know, with respect to the input, right, that, that can be much larger than the 3 cross, immediate 3 cross 3 that it is seeing, because each layer in turn, that neuron is seeing something else with the input and yesterday we saw that example, right, so receptive field. So this could happen at any layer for that matter, right. So here we are just considering this to be the, to be the first input going in, but you know, it will generally apply, I mean, wherever you are, right. And therefore, right, what you find is that, you know, you are saving on the parameters, right, instead of, instead of say 21, right, now you have got only, only say, right, 9 to, 9 to find, which means that the parameter sharing has not yet happened, right.

What this means is that you still have, you still have, right, 3 parameters for this, 3 parameters for this and 3 parameters for this, that you are independently finding out, right. So the weights are still like 3 into 3, just a toy example, but then, right, it is storage efficient because now we have to store fewer weights and then write runtime also, it says faster and then if you go for parameter sharing, right, so what you had was this situation, right, where the earlier one was like you had different weights coming for each neuron. But as I said yesterday, when you talk about a convolution, you sort of do not just assume that, of course, in your 1D, right, you would have seen, you know, things like impulse response being, you know, infinite in support and all that, that is also possible. But those are not the situations that we kind of deal with in, in a kind of network. We simply limit it to something like, let us say FIR, right, a very finite, finite sort of, you know, support and in fact very local, okay, and that suffices, I mean, there is no reason to make it, make it very big.

And there is one way, right, by which you can still, you know, if you really feel that you should be looking elsewhere in an image and it is not enough to just look at locally, there is always something called an attention that can be brought in, okay, which will then kind of focus on things that are really relevant, okay. So that is a mechanism that came in later and people figured out that that is a much better way to sort of, you know, to sort of what you say, to sort of, you know, use the information in image than to increase the depth by L, by L, you know, by increasing the number of layers. That is one way by which you can get more and more exposure to the image, but the better thing to do or more efficient thing to do is to use what is called attention. If time permits, I will just maybe talk about 10, 15 minutes about that, but let us see, right, how it goes. And if it is parameter sharing, then we simply say that they all share the same set.

So W1, W2, W3, this is like really a convolution now, right, because you are kind of translational invariant and you are very local and this is like convolution with a very, very finite sort of a support, right. And so, right, this is even more efficient in terms of the weight storage and so on. And one of the things that you will kind of notice is that, right, between an MLP and actually a CNN, you see that, you know, you already have very, very few weights, I mean, right, therefore a regularization sort of implicit, right, because, you know, you do not allow for too many of these unknowns to come in and so on, instead of playing with just few parameters and you are hoping that they will actually do the job, unlike an MLP where you had to, but even then it does not mean that you do not use regularization, but required to much lesser extent because the parameters to begin with are not really that many. Actually, we will see a network, right, that will actually give you a better idea. Then you can actually extend it to the case of more input channels, right, like I said, this could be at any layer and then, right, if you have more input channels and if you are also sharing the weights, then what you see is that, right, these weights will be the same, right, across the neurons, that is the parameter sharing.

They are local because of the fact that the support is still like, you know, fixated on just 3 in the input channel 1 and 3 in the channel 2, but what you find is that the weights that go and then work on channel 2 are not the same as the weights that go and work, that go and work on channel 1. This is exactly what I said yesterday, right, if you had multiple channels and if you are having a box filter, then the filter that acts on the first channel can be, say, different, the weights, right, can be different from what is acting on the second channel, can be, you know, different from, see, finally a network might figure out that maybe the same set of weights will work, but we do not force it. And of course, because of parameter sharing, so this is exactly what I had shown yesterday, right, so you could have something like this and, and that you have, okay, I have a different, I need a different color, but anyway, so you have a box filter that runs through and through and that box filter, right, will have a set of weights for this, will have a set of weights for this, will have a set of weights for this, but overall it is just one filter, okay. This is for, this is for, let us say, one channel, right, this is for one kind of a feature map and then you can again change this box filter, right, when you want, when you want, you know, a different, a different output and so on, one more channel if you want, then you actually change that box filter, change in the sense that the weights and all will again change, you have a different box filter that then acts on that volume. So this is for the case of two input channels, right, and so I think, you know, this is something that you saw yesterday, but not in this form, right.

Then you can also have one input and then two output channels, right, this is exactly what I was talking about right now. So, so for example, right, so, so in this case what will happen, you have a parameter sharing. So for example, these weights, these weights are all, are all, are all the same, right, which is like, which is like saying that, saying that I have this and then I have, so this is like, okay, so I need, I need to see two output channels, I have one input channel, right. So what I will do is I will have one filter that works on this and then gives me this, this output and then, and then I have got another filter that again works on the same input gives me, gives me, gives me a different feature map, that is what you are seeing here. So the second output channel is this, right, and these weights are again shared by which we mean that as you go from here to here to here, you keep on applying a convolution, then whatever values you get here, they are all obtained with the same, same filter, you do not change the filter, right, that is why those colors are all the same, right, these colors from, from neuron to neuron within a layer.

So, so when you look at this, you should, you should look at it as kind of neuron sitting in this spatial grid. When you look at this, you should look at it as neuron sitting at the input. This is 1D, but in 2D, that is how, that is how you would actually visualize, right. I mean, 1D maybe sometimes is easy to visualize, but 2D is also equally convenient, right, and that is, that is what, this is what we saw yesterday, but this is just a different way of visualizing the same thing. Then a generic level, right, looks like this.

So there is a local connectivity, right, which is what, by which we mean that there is a spatial support which is very, very finite and limited in its spatial extent. And by the way, the weights and all, we are not forcing anything, right, these are all of course, of course, the real weights, I mean, people have tried complex CNNs and all, but I do not think much has come out of that. So really, weights are all real, they can be negative, positive, right, we do not put any constraints on them. And what is happening is at any, at any, for example, this is an output map. So, so if you are, if you are looking at, looking at any one feature map in the output, right, then you are really looking at a box filter, okay, that is locally acting through the, through the right input channels.

So an input could be like multiple channels that you have here and you have a box filter that is kind of working through the, through that volume, right, and whatever is that. So this cone, it will kind of, you know, eventually lead to 1 pixel here, then, then maybe right after this you are doing something and then, and then you have got, you have got one more feature map out there, then that will kind of work through this volume, right. So you will have like this volume that it will work through and, and then all of this will kind of say converge to a point here. You may do something in between also, it does not mean that convolution has to be followed by a, by another convolution, you can do something inside, between the two, but whatever it is, right, so you are sort of, right, this is what you are effectively doing. So you are doing parameter sharing, you are doing local sort of a connectivity, right.

Now in general, if you look at a CNN architecture, right, any CNN for that matter, you will typically find these kind of things, okay. So one is a convolutional operation, which we have already seen, and then there should be a nonlinear activation because convolution is all the linear part, right, where you are just, just doing your weighted averaging, adding a bias. I am not so explicitly speaking about bias, but it is all there. And what you are, what you are really doing is, right, you are doing convolution followed by a nonlinear activation and typically that will be reloop. And then there is a pooling operation.

Pooling operation is typically done, you know, for two reasons. One is you might just want to reduce the size of your feature maps. It is, you can think of it, it is not exactly a sub-sampling operation because sub-sampling typically means that, you know, for example, you know, if you have four elements, right, you can just merge them to one, okay, that is typically sub-sampling. You could just uniformly average them or you could have a filter that actually tells how each one should be weighted. So, in a simplest way you just average them, but that is called, that is a way to do, that is a pooling, that is called average pooling, but there is something called max pooling where you actually pick the one that is maximum, right, in that and the idea is that take the maximum information and not really propagate, right, everything out there.

This was, this has been originally proposed in other, in the guise of something else, okay, and I think the CNN folks gave it a name as max pooling, but this was not a very new idea. I mean, so the people have used this before. I do not know what terminology they were using at that time, I cannot recollect now, but you know, max pooling is a name that came later. Then what you could typically have is again a convolutional railing and, okay, this is not kind of say necessary that you will always have all of this. Sometimes you may have a pooling, you may not have a pooling and that is all sort of, you know, a trick of the trade, right, in the sense that what architecture works best for a problem is something that nobody knows, okay.

Sometimes you take cues from architectures that have already been built for certain problems and then you know that that may be a good starting point, right, instead of just starting from, you know, in the wild. So that can be a good starting point, but no one really knows what is the best way to kind of go forward. And what you typically, if you have a classification kind of a problem, not kind of say necessary that every classification should end up in a fully connected, you know, fully connected output layer, but many a time you will find that, you know, there will be an FC layer which is fully connected, a bunch of FC layers and in this case what is happening is you have come till this point and these are all still your 2D feature maps and you are sort of unwrapping them, right. So that is called the flattening part. You can flatten in whichever way you want.

It is called lexicographical flattening. So you may put like, you know, first row, I mean, you can unwrap in whichever way you want. Even network will figure out accordingly. If you unwrap the other way, if you take this way and then unwrap or if you take this way and then unwrap, whichever way you unwrap, it will then figure out what is the best way to learn the weights for that unwrapping. But eventually the length of the vector will be the same, right.

So this one-dimensional vector that you get, right, that will be the volume of, so right, if you think about this m and k, so it will have a dimension m into n into k, right. That will be the dimension of that one-dimensional vector. So that is called a flattening operation. And then after that you will get these, get a next fully connected layer if need be. Typically it will be there and then maybe, right, you have a classifier, you know, the output layer with a softmax sitting there and then that is going to do a classification.

Oh, excuse me sir. Yeah. Sir, why are we using a ReLU in the process? Why are we using a ReLU? No, I mean, you know, for the same reason that why you need a non-linear activation. Why should one not use ReLU? No, sir. If I do not use any activation. No, no, but then the point is a non-linearity, right, you have to bring in, no. ReLU brings in a non-linear activation.

Otherwise all the operations will become linear, right. Otherwise you can combine two convolution blocks and effectively you get only one filter, right. So, this non-linearity, see, the whole thing behind neural networks is this kind of non-linearity that you are trying to bring in, right. And the non-linearity you can bring in only through some activation.

It can be sigmoid, it can be something. But as I said, right, ReLU, the gradient problem is not there with ReLU. Even if x is high, right, I mean gradient is still 1. Whereas sigmoid and all you typically do not use at all. In fact, right, sigmoid and all actually ends up being used in, you know, RNNs, recurrent neural networks when you want to do a gating and so on, which I think next class I will talk about, right. So, really sigmoid and all, you know, you do not even use it that much, like even softmax is what you use at the end, right, typically for a, you know, multi-class problem.

So yeah, so I think, you know, you could have used anything else for that matter, leaky ReLU you can use. But typically you need a non-linear activation there because that is what will then pull out interesting features which you then aggregate over channels, right. And if you are asking why are you kind of using a box filter, the answer is still that you have to aggregate information coming from various feature maps. And what is relevant and what is not will be up to the weights that you find, right. If you find that, you know, the network does not believe that the last feature map is really useful, just put all of them as 0 or whatever or very low values, not 0, it will give just very low weightage to them.

Okay, now, right, just wanted to play out these animations, right. All of you are familiar with the convolution operation, but just at a 2D, right, as I thought, if you have not seen this before, so it is like a filter, right. So the yellow thing that you see, 1, the cross, that is the filter, it is like 101, 010, 101, right. And also one more thing, right, by the way, I mean, do not get worried if you do not see a correlation versus a convolution sort of, you know, a distinction. Like for example, a correlation is like what y of n is summation, what is that, xn, hn plus m, right.

That is what we do in 1D. But in, but here people do not bother so much about, you know, flipping and like I said yesterday, right, ideally in convolution you should flip about both the x and y axis and that is when you get a filter. But then, you know, all that network will kind of figure out inside. If you do not flip then, you know, then it will kind of understand it accordingly and move on, right. So really do not get too worried, I mean, if somebody is not paying too much attention to that flipping and all. Of course, if you have a symmetric case, then there is no issue at all, right.

I mean, if it is isotropic, there is no issue. But if it is not, you know, then it will learn accordingly, right. So you see, so you see that, right, you are taking this kernel and you are shifting it, right, just like you do in 1D, right. I mean, you flip it and then you get a shifted left, right, top, bottom. Here you have to go like left, right, top, bottom, right, all over you have to scan and then everywhere you take a weighted sum. For example, in the last place, right, what do you get? What was it like 1 plus 1 plus 1 and then 1, 3 and then 4, right.

So the last entry should be 4, right. So whatever, right, so it is kind of, so it is easy to figure out. So the last one, right, I was just talking about this entry, right. So wherever you are, right, I mean, you can do this. Now if you, there is one more thing, right, people sometimes use what is called a zero padding, okay. This is to sort of, if you feel that, you know, for some reason you want to, you want to sort of play around with the size of the feature map.

Because you know, see one of the things that you will realize is that when you are doing a convolution operation, right, this is your image on which you are trying to do a convolution and I mean, if it is a 3 cross 3 kernel, so what I am saying is that, so if you have a kernel sitting here, right, now you will, you will not be able to take that kernel here, right, because then it will kind of spill over, right, so you will get something like that, right, I mean, so one row and one column will sort of go out of the image, right. Therefore, you may think that I have to stop inside. If you stop inside, then effectively your size has gone down. This is somewhat counter to what you typically do in 1D, right, in 1D it is like, right, m plus n minus 1, right, I mean, that is the eventual size, that is not the way we do it here, okay. Now what typically is done is you kind of zero pad, okay, so if you want to play around with the size, you just zero pad.

So what you do is, so for example, in this case without padding, right, so the way you have to interpret it is, right, this is the kernel, that 3 cross 3 thing that is going around, that is your filter and this is your input, right, and then output is here, right, so what you see is that here if you just let it, every time you just let it stay inside the boundary and therefore, right, you can only get, you see, 4 values which is like a 2 cross 2 output map. So there is another thing called stride which is like how much jump you do, okay, and stride, again, right, something that can be, that is typically used along with a convolutional operation. So you can see here that when you are moving, so first thing to notice is what is the size of the filter, right, and then second thing that you should notice is what is the size of the input feature map that you have and accordingly, then look at whether you have zero padding and whether you are using a stride. Now here stride is 1 means you just go like 1 step at a time, you do not jump over, whereas stride to, right, you see, for example, right, I mean, first I will start here, then I will jump there, okay, then from here I will jump here, right, because I have to jump at a rate, if you see here, right, it will jump there and then it will jump there, right.

So it is like a stride of 2. So stride means both along X and Y are jumping, right. So that is, you can have stride whatever, right, does not limit, these are small ones, so we are taking 1 and 2, but you can have stride 4, 5 and so on. And again, accordingly, what size you get here will change, okay, but there is a systematic way to arrive at that number, okay, what should be the output, right. Then you can have both zero padding and stride. So here is a 3 cross 3 kernel that is being moved around, right, and you are trying to convolve with the input and then output is out here and then you can add zero pad and a stride of 2.

So 2, 2 means along X2 and along Y2. Typically, this is all, I mean, uniform along X and Y, but there is no reason to force them that way. So for example, right, so the way to kind of, okay, think about it is this. So if your input feature map is like, say, W1, okay, here it is, right. So if you are accepting an input of size W1 cross H1 cross D1, what is D1? The number of feature maps, right, number of feature maps in the input, W1 cross H1 is the spatial extent, okay. So if you have W1 cross H1 and then D1 number of such feature maps and if you are using number of filters k, what does that mean? That means in the output you will have k feature maps, right.

So I mean all this now you should be kind of comfortable with. So when we say number of filters k, that means we are trying to use k number of box filters and if you use k number of box filters, you should actually get k output feature maps, right. And spatial extent F, so spatial extent F means of the filter that you are using. So here we are assuming it to be F cross F, but it can be F1 cross F2 also. There are situations where you need a rectangular kernel and all, okay, that generally F cross F, stride is S which you know now and then amount of zero padding is P.

Then what you have is W1 minus F plus 2P by S, so it is like this, okay. Whether the way it is written is a little confusing, so think about this as W2 is equal to W1 minus F plus 2P by S plus 1, okay. And similarly H2 is whatever, H1 minus F plus 2P by S plus 1 and D2 which will be the, which is what is the, what is the depth of the output feature map, that will be of course equal to k because that is what, that is the number of filters that you are using, right. And so of course you can calculate how many weights and biases that I think yesterday itself I told you, right. So in this case how many unknowns will you have? So for example, right, I mean you have a filter of size F cross F, then you have what, you have a D1 size, right, in the input.

That means any variety you have to find for one box filter you should find F into F into D1, right. And then you have got k, what is it, number of filters k, right. So into k that many feature maps, so that means that many filters, box filters, k number of box filters and then plus the biases which will be like k, right, because and always remember, right, for one feature map there is only one bias. Just like you have spatial invariance of the weights, the bias is also invariant, right. So it remains a constant across all the neurons within a feature map, okay.

So it is like plus k. And therefore, right, and of course, right, so the pooling operation, right, looks like this. So, so right if you have 1, 1, 5, 6, right, so and if you say that I am, and pooling also you can, you can get a say define in terms of the filter. The filter simply means what is the local extent that you are seeing. And there is no filter to be estimated here, okay. This is unlike the previous things where you have to estimate weights.

There is no, there are no unknowns that a pooling layer introduces. It does not introduce any unknowns. Just a matter of whether I take the support like 3 cross 3 or whether I take 2 cross 2 or whether I take 2 cross 3, what size I want to take. So here the filter says 2 cross 2. That means you are just looking at a 2 cross 2 region. And then whatever is the maximum value you kind of, you kind of put that.

And similarly you take this, find that the maximum value is this, 8, put that and whatever it here it is 4, so you put that here, okay. And here also you can have a stride. So which basically means that I can start here. I could have taken this one also if my stride was 1. I could go like you know one step at a time in which case, in which case you know I will get basically one more entry here, okay.

But in this case the stride is actually 2. So I am taking here, then I am jumping, okay. Then I am jumping down, then I am jumping down. And the same formula applies, right, whatever you had there. And the main reason why you want to sort of you know use the, use you know pooling is to actually reduce the spatial, you know is to make sure that you have fewer unknowns and also the fact that it will probably help you learn, you know learn in a sort of you know a better manner and then computationally that it will be advantageous if you have fewer unknowns to estimate down the line, okay.

And always remember pooling does not involve any new computation. Therefore, when people talk about layers, pooling is not considered as a layer because layers typically mean that some sort of a computation should happen there. That means some weights have to be estimated, something should go on there. This is a typical terminology, right. So if you see that you know something has 6 layers, then you will actually that will typically mean that there are 6 layers where the computation is going on. So in between if you have 4 pooling, I mean it would not add to the number of layers.

I mean this is generally the case but if somebody you know represents it in a different way, this is the thing. Of course average pooling can also be used but not as popular as max pooling, okay. And the formula is exactly the same, right.

You can cross check all this, right. I do not want to do this. Please go ahead and cross check, okay. And so the ReLU anyway, right, you people know flattening, softmax, right, all this you know, okay. So I think let me go to the next one, right. So I thought what we will do is you know we will take one architecture today, right. That is what I wanted to do and AlexNet, right, for example, okay, which is one of the most important sort of architecture, which is one of the very first, right, to actually emerge, okay.

And I thought we will do that today, okay. So that right, I mean one architecture if you understand, I mean you know you will get a, this one we will go in, yeah, I mean not in great detail but still to fully convolutional. Does that mean that it should flatten the entire image and do only convolutional? No, no, no, fully convolutional means 2D convolution. Fully it is actually 2D, okay. So nowhere any flattening happens.

So it is like till the last one you have a 2D feature map. That is fully convolutional. So nowhere are you doing of an NFC layer fully connected. We will have like m into n number of eights. That will never happen.