11.10.2020

NEURAL QUBITS

KTHack2020 - KUANTUM TEKNOLOJİLERİ HACKATHONU

Takım Üyeleri:

⟨ ALARA | ZINDANCIOĞLU ⟩
⟨ BAŞAK | EKINCI ⟩
⟨ ENES | BAŞPINAR ⟩
⟨ TOGAN | TLIMAKHOV | YUSUF ⟩

0) İçerik

1) Özet

Makine öğrenmesi, bilgisayar bilimi ve mühendislikteki en büyük araştırma alanlarından birisidir. Askeri, endüstri, güvenlik, robotik ve astronomi gibi bir çok alanda çok önem arz eden bir teknolojidir. Öte yandan kuantum programlama ise insanların hayatına çok kısa bir süre önce giriş yapan yeni bir alandır. Yapılan öngörülere göre klasik bilgisayarlardaki transistörlerin boyutu daha fazla küçültülemeyeceği için gelişim kısa bir zaman içinde mümkün olmayan bir hale gelecek ve bu yüzden de kuantum bilgisayarlar ön plana çıktı.

Bunun sonucunda insanlar, klasik bilgisayarların kullanıldığı uygulamaların hangilerinde kuantum bilgisayarların daha verimli olabileceğini araştırmaya başladı. Bir çok alanda yapılan bu araştırmalar, elbette makine öğrenmesi alanına da sıçradı. Dünyanın dört bir yanında insanlar farklı modellerin, yöntemlerin ve algoritmaların kuantum programlamaya aktarmaya çalıştı ve performanslarını değerlendirdi. Bazı noktalarda ise klasik bilgisayarlarda yapılan hesaplama tekniklerine göre performans ve hız açısından daha verimli olduğu keşfedildi.

Biz de hackathon konumuzu bu sebeplerden dolayı, kuantum programlama içerisindeki önemi üst sıralarda yer alan makine öğrenmesi alanında seçtik. Projemize küçük adımlarla başladık. Öncelikle Neural Network'ü kuantum programlamaya uyarladık (QNN). Sonrasında modelimizi, verilerimizin kuantum devrelerine çevrilmesi aşamasına katkı sağlayacak ve klasik sinyal ve görüntü işlemede kullanılan en önemli algoritmalardan biri olan Fourier algoritmasıyla zenginleştirdik. Bunları yaparken edindiğimiz bilgiler ile projemizdeki en önemli noktası olan Convolutional Neural Network modelini, kuantum programlama için sıfırdan inşa ettik (QCNN) ve hackathonu tamamladık.

2) Proje Detayları

Projenin başında ekipçe kuantum programlama ile ilgili bilgilerimiz çok yüzeyseldi. Bu yüzden, öncelikle kuantum programlamaya dair eksiklerimizi kapatarak başladık.

Öncelikle basit bir Quantum Neural Network (QNN) oluşturmaya karar verdik. Bu kısımda karşımıza çıkan ilk aşama verilerin kübitler şekline dönüştürülmesi oldu. Bildiğimiz üzere kübitlerin başlangıç değerleri sıfırdır. Kübitler ile pikselleri ifade edebilmemiz için resimlerdeki piksellerin değerlerini eşik değerlere göre ikili (0 ve 1) hale dönüştürdük. MNIST verisetindeki resimler 28x28 olduğu için 784 kübite ihtiyacımız olacaktı. Ancak kuantum bilgisayarı simülatörlerinde mümkün olduğunca az kübit kullanmamız gerektiği için resimlerin, diğer değerleri de denedikten sonra, 3x3 piksele düşürmemizin en iyisi olacağına karar verdik. Tensorflow'daki resim küçültme işlemi piksellerin ortalamalarını alarak sıkıştırma işlemi yaptığı için sınıflandırma yaparken modelimizin performansını klasiğe göre düşürse de çok fazla problem yaşatmadı. Ve bu sayede her resmi 9 kübit ile saklayabilecektik. Bit değerlerini kübitlere aktarmak için resim başına bir tane devre oluşturduk. Artık her bir devremizin sonucu bizim için bir resmi ifade edecek hale geldi. Ardından modelimizi oluşturduk ve Keras içerisinde bulunan PQC katmanını ekledik. Bu katman, modele verdiğimiz resmi işleyen ve sonucu readout isimli kübite aktarma işlemini gerçekleştirir. Modelde kullandığımız Hinge kayıp fonksiyonu sayesinde tahminimiz -1 veya 1 değerini alır. Başlangıçta ise -1 değerini 6 rakamının sınıfını, 1 değerini 3 rakamının sınıfını ifade edecek şekilde belirlediğimizden dolayı, tahmin yaparken bunu göz önünde bulundurduk. Son olarak tahminlerimizi gerçekleştirdik. Modelimizin eğitimi 29 saniye sürdü. Temel parametrelerden epoch=3, batch-size=32 ayarlandığında en yüksek doğruluk olan ~0.83 değeri elde edildi.

Çalışmamızın bir sonraki aşamasında QNN algoritmasında fourier dönüşümü kullanarak, bu algoritmayı geliştirmeye çalıştık. Fourier dönüşümü, sinyal ve görüntü işlemede kullanılan en önemli algoritmalardan biri iken, aynı zamanda modern kuantum algoritmalarının çoğunda da önemli bir bileşen olarak kabul edilmektedir. Literatürde yapılan çalışmaları incelediğimizde, kuantum hesaplamalarda görüntüleri kübit olarak sunmanın bir kaç yolu bulunduğunun gördük. Bunlardan Flexible Representation of quantum images olarak adlandırılan, fourier dönüşümünü temel alarak görüntüleri kübit olarak sunmamıza yarayan yöntemi kullandık. 3x3 ve 4x4 boyutlu resimlere ayrı ayrı bu yöntemi uygulayarak sonuçlarımızın sadece qnn algoritmasın kullnılmasıyla elde edilen sonuçlarla hemen hemen aynı oranda doğrulukla elde ettiğimizde gördük, ileride bu yöntemi nasıl iyileştireceğimiz yönünde çalışmalar yapmaya devam etmek istiyoruz.

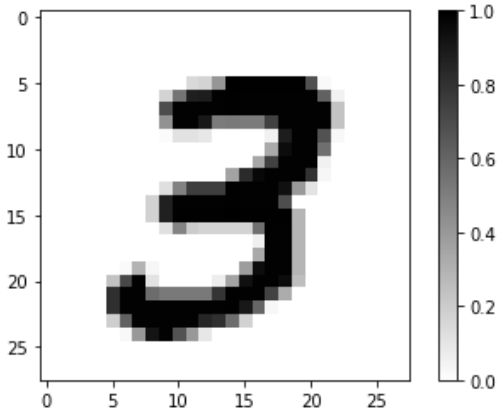
QNN modelinde olgunlaştıktan sonra resim sınıflandırmada çokça kullanılan Convolutional Neural Network (CNN) modelini kuantum programlama ile inşa etmeyi planladık. Araştırdığımız makalelerden gördüğümüz kadarıyla inşa edilen Quantum Convolutional Network (QCNN) modellerinde, kuantum algoritmaları ile yazılan convolutional filtrelerin resmin ön işlemesini yapmak için ve yahut klasik CNN modelleri ile birleştirilerek eğitildiğini gördük. Resimleri QCNN algoritmasına girdi olarak verebilmemiz için, her piksel değerini kuantum durumuna dönüştürmemiz gerekiyordu. QNN modelinde olduğu gibi, resimlerin piksel sayısının çok fazla olması kuantum modelinin eğitilmesine izin vermeyeceği için verisetimizdeki tüm resimleri 6x6'lık piksele indirip resimlerin ortasındaki 4x4'lük kısmı kullandık. Ve her resim için 16 kübitlik devre kurduk. İlk modelimizde bu resimleri, 1x2 boyutundaki Quantum Convolutional filtreden geçirdik. Bu model de MNIST verisetindeki 3 ve 6 sayılarında sınıflandırma yapılarak test edildi. Eğitim için her birinden 100 resim, test için ise 20 resim kullanıldı. Eğitimi sonucunda ise %78 doğruluk oranına ulaştık. Geliştirdiğimiz diğer bir QCNN modelinde ise yine 4x4'lük resimler kullanıp bu sefer 2x2 boyutunda Convolutional filtreler uyguladık ve iki katmanlık bir model oluşturduk. Yine MNIST verisetindeki 3 ve 6 sayılarında eğitip test ettiğimiz modelimiz %80 doğruluk oranına ulaştı. QCNN modelini tamamen kuantum programlama ile ifade etmemiz, projede elde ettiğimiz en önemli kazanım oldu.

3) Proje Aşamaları

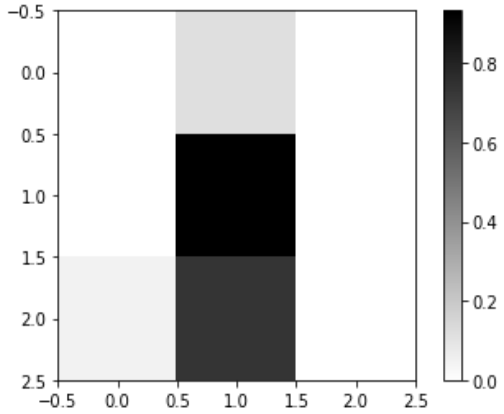
3.1) Aşama 1: Quantum Neural Network (QNN) Modeli Oluşturulması

Hackathon proje konusu olarak belirlediğimiz kuantum makine öğrenmesinde oluşturduğumuz ilk model Quantum Neural Network (QNN) oldu. Bu modelde kullandığımız yöntemler, Tensorflow'un kuantum modülü ile Python'da implemente edildi.

Modelimizi oluşturmadan önce yaptığımız ilk şey, MNIST veri setindeki verileri kuantum programlamaya uygun duruma getirmek oldu. Öncelikle verilerimizi ikili sınıflandırmak yapmak üzere 2 sınıfa indirgedik (3 ve 6 rakamı) ve bu filtreyi eğitim ve test listelerine uyguladık. Ardından resimlerimizi kübitlerle ifade edecek olmamızdan ve bazı kısıtlardan dolayı kübit sayısını mümkün olduğunca küçük belirtmemiz gerektiğinden, orjinal hali 28x28 piksel olan resimlerimizi 3x3 piksel boyutuna indirdik:

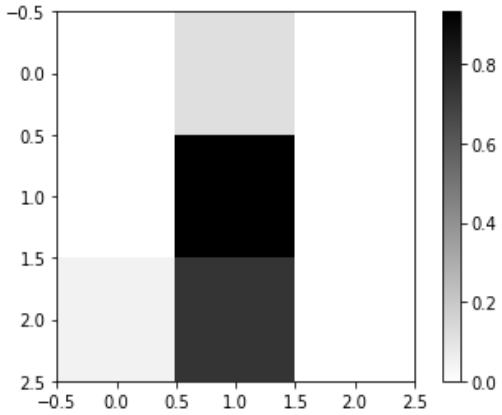


(Görüntünün orjinal hali, 28x28 piksel)

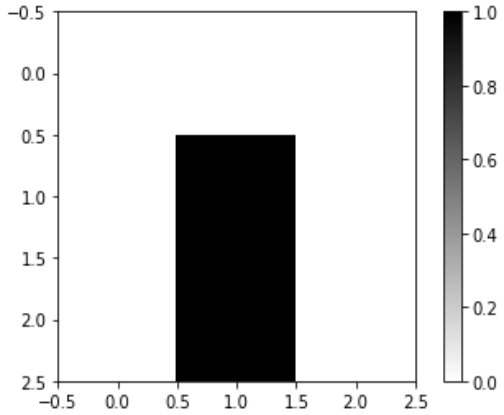


(Görüntünün ikili kodlamadan sonraki hali, 3x3 piksel)

Boyut düşürme işlemin sonucu anlamsız görünse de aslında yeni oluşan pikseller öncekinin özelliklerini taşırlar. Bu sayede, sınıflandırma yaparken yararlanabiliriz. İşlem sonucunda resimlerimizi toplamda 784 piksel yerine 9 piksel gibi küçük bir alanda depolayabildik. Bu işlemden sonra resim piksellerinin değerlerini belirli bir eşik değerini baz alarak 0 ve 1 değerlerini alacak şekilde ikili kodlamaya dönüştürdük:



(3x3 görüntünün orjinal hali, piksel değerleri 0-255)

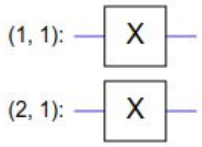


(3x3 görüntünün ikili kodlamadan sonraki hali, piksel değerleri 0-1)

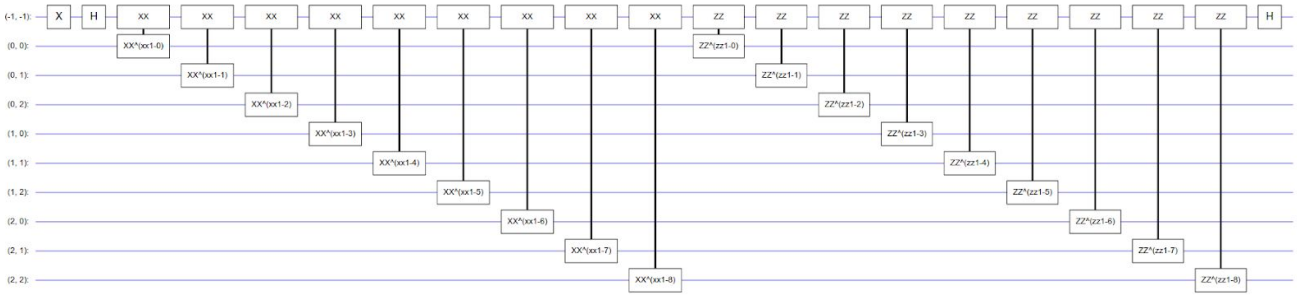
Sonrasında her bir resim için, yine resimleri ifade etmekte kullandığımız 16 kübit oluşturduk. Kübitlerimizi de şu şekilde görselleştirecek olursak daha anlaşılır olacaktır:

```
cirq.GridQubit(0, 0), cirq.GridQubit(0, 1), cirq.GridQubit(0, 2)
cirq.GridQubit(1, 0), cirq.GridQubit(1, 1), cirq.GridQubit(1, 2)
cirq.GridQubit(2, 0), cirq.GridQubit(2, 2), cirq.GridQubit(2, 2)
```

Kuantum programlamada kübitlerimiz başlangıçta 0 değerini alır. Resmimizdeki değeri 1 olan pikselleri kübitlere aktarmak için X kapılarını uygularız. Örnekteki resmimiz için devremiz şu şekilde olacaktır:



Ardından model devremizi oluşturduk:

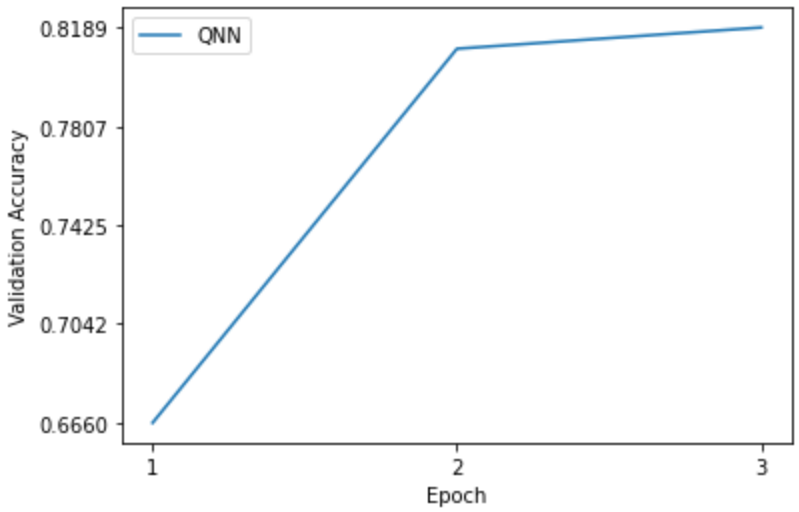


Keras mimarisini kullanarak kuantum bileşenlerini kullanacak olan modelimizin kuantum devremizi oluşturduk. Model devresini kuantum verileri üzerinde eğitmek için “tfq.layers.PQC” (Parametreli Kuantum Devresi Katmanı) kullanır. PQC katmanı, resmimizin bir sınıfa ait olma olasılığını “readout” kapısı kullanarak [-1,1] aralığında döndürür. Dolayısıyla verilerimizin eğitim ve testte kullanacağımız verilerin 0 ve 1 olan etiketlerini de, -1 ve 1’e dönüştürdük.

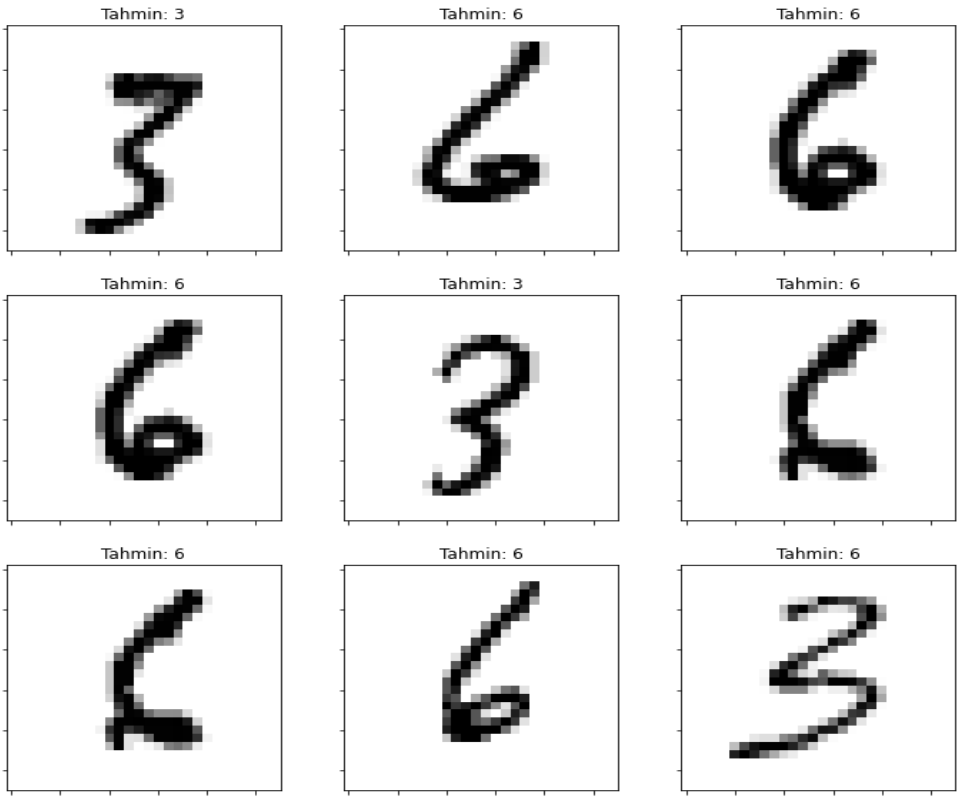
Gerçekleştireceğimiz ikili sınıflandırma işlemi için oluşturduğumuz modelin özeti şu şekildedir:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
pqc (PQC)                    (None, 1)                 18
=================================================================
Total params: 18
Trainable params: 18
Non-trainable params: 0
_________________________________________________________________
None
```

Modeli eğittimizde ise en iyi sonuç 3x3 piksel resimlerde, 3 epoch, 32 batch-size parametreleriyle yapılan ve 29 saniye süren eğitimde ~ 0.83 doğruluk oranıyla alınmıştır:



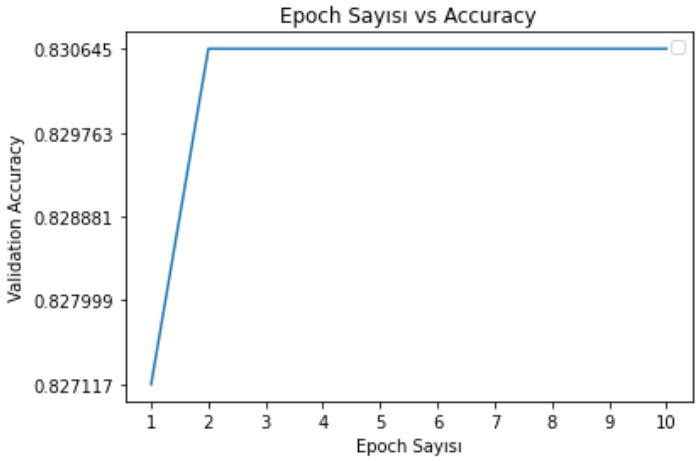
Model ile yapılan bazı tahminler aşağıdadır:



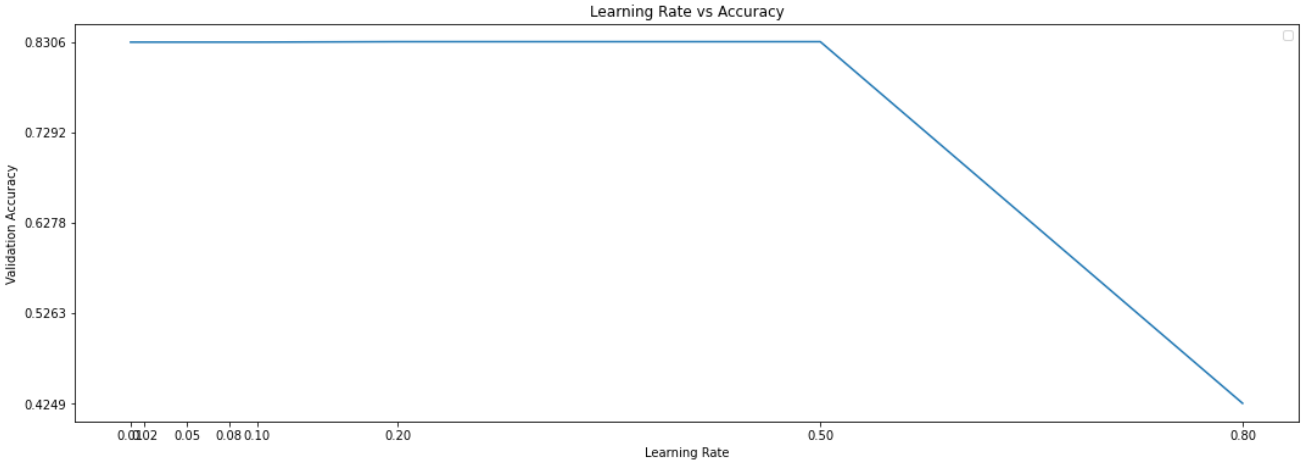
3.1.1) Performans

Modelin performansı değerlendirilirken farklı parametrelerde kullanılan farklı değerlerin grafikleri çıkarılmıştır.

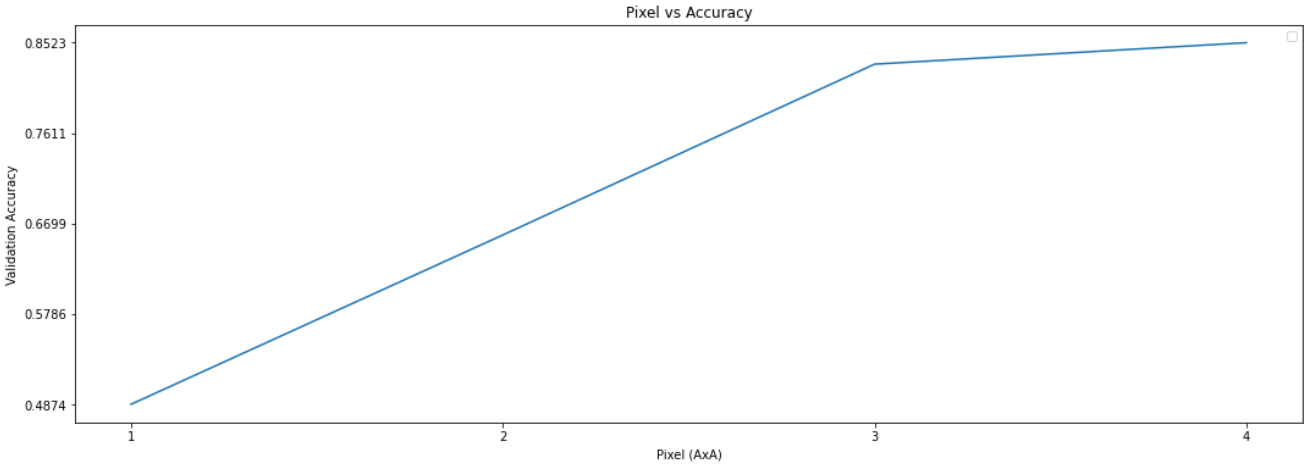
İlk ele aldığımız parametre epoch sayısı. Model 1-10 epoch sayısı aralığında ~16.000 veri ile batch_size 32 olacak şekilde 10 kez eğitilmiş ve aşağıdaki sonuç alınmıştır. Epoch sayısı 2 ye kadar doğruluğu arttırır iken 2 den sonra sonuca etki etmemiştir.



Ele aldığımız ikinci parametre learning rate. Model, [0.01, 0.02, 0.05, 0.08, 0.1, 0.2, 0.5, 0.8] learning rate ile ~16.000 veri ile eğitilmiş ve aşağıdaki sonuç alınmıştır. Learning rate 0.50'ye aynı sonucu vermiş, sonrasında model güzel bir şekilde yakınsamamaya başlamıştır.



Ele aldığımız üçüncü parametre resim piksellerinin büyüklüğü. Piksel sayısı, oluşturmamız gereken kübit sayısını da etkiler. Model 1x1, 2x2, 3x3, 4x4 piksel olmak üzere üç kez eğitilmiş ve aşağıdaki sonuç alınmıştır. 2x2 ve 3x3 arasındaki süre performansa göre kısa iken, aynı oran 3x3 ve 4x4 arasında uçurum seviyededir. 0.02'lik bir artış için 1937 saniye fazla zaman harcanmıştır. Bu sebeple en makul değer 3x3 pikseldir.




```
[1x1: 8 saniye, 0.4874] [2x2: 15 saniye, 0.6583]
[3x3: 43 saniye, 0.8301] [4x4: 1980 saniye 0.8523]
```

3.2) Aşama 2: Kuantum Fourier Dönüşümünün QNN Modeline Aktarılması

Üzerinde çalıştığımız QNN algoritmasında fourier dönüşümünü kullanarak, bu kodu geliştirmeye çalıştık.

Fourier dönüşümü makine öğrenmesinde, karmaşık olan bir verinin boyutlarının azaltılarak daha basit bir problem olarak incelenmesi için olanak sağlamaktadır. Fourier dönüşümü, sinyal ve görüntü işlemede kullanılan en önemli algoritmalardan biri iken, aynı zamanda modern kuantum algoritmalarının çoğunda da önemli bir bileşen olarak kabul edilmektedir.

Kuantum fourier dönüşümünde girilen data, olasılık genliği olarak algılandığından klasik fourier dönüşümünden farklıdır. Bu durum ilgili discrete Fourier transform (DFT)'da genliklerin değişmesine neden olmaktadır. Klasik durumda kullanılan algoritma, bütün kompleks vektörleri alıp tüm DFT'yi aynı uzunlukta başka bir vektöre çevirmektedir.. Yapılan çalışmalarda, bu işlemelerin kuantum bilgisayarlarda, klasik bilgisayarlardan daha hızlı gerçekleştiği görülmüştür[1].

Quantum hesaplamada görüntüleri kübit olarak sunmanın bir kaç yolu bulunmaktadır. Bu yöntemlerden bazıları FRQI (Flexible Representation of Quantum Images) [2] ve A novel enhanced quantum representation (NEQR) [3] algoritmalarıdır. Bu algoritmalar, görüntülerin az sayıda kübit kullanarak sunulmasını sağlayarak, kullanılan algoritmada görüntülerie uygulanan kapıların sayısının daha az olmasını sağlamaktadır. Fakat bu yöntemlerde kullanılan görüntü sayısı arttığında beklenilen performans elde edilememektedir[4].

Çalışmamızda, FRQI yöntemini kullandık, bu yöntem 2^n x 2^n lik bir görüntüyü toplamda 2n+1 kübit olacak şekilde tutmaktadır.

FRQI yönteminde görüntü, $$\left| I(\theta) >= \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (\cos\theta_i \,|0> + \sin\theta_i |\,1>) \otimes \right| i >$$

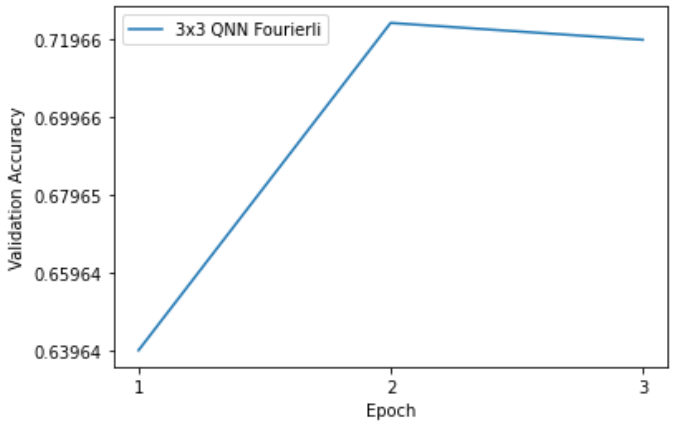
[1]

ile verilen |I(θ)> durumunda tutulmaktadır. Burada theta görüntünün rengini kodlarken, |I> durumu, görselde piksellerin konumunu kodlamaktadır.

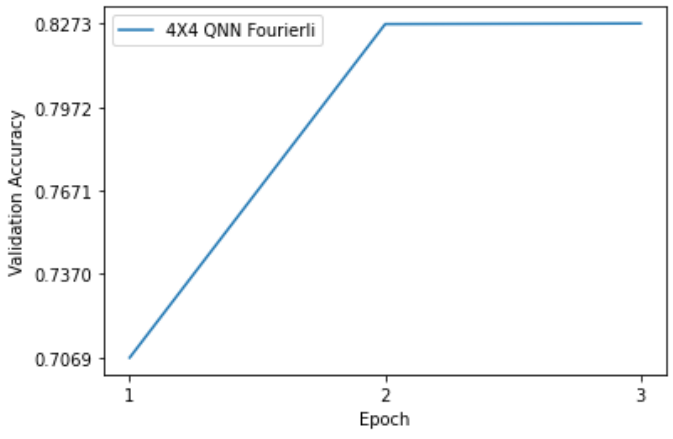
FRQI yönteminde $2n$ kez Hadamard kapısı uygulanırken, 2^{2n} kez kontrollü dönmeler uygulanmaktadır. Bu kontrollü dönmeler CY ve NOT operasyonları ile gerçekleşmektedir.

3.2.1) Performans

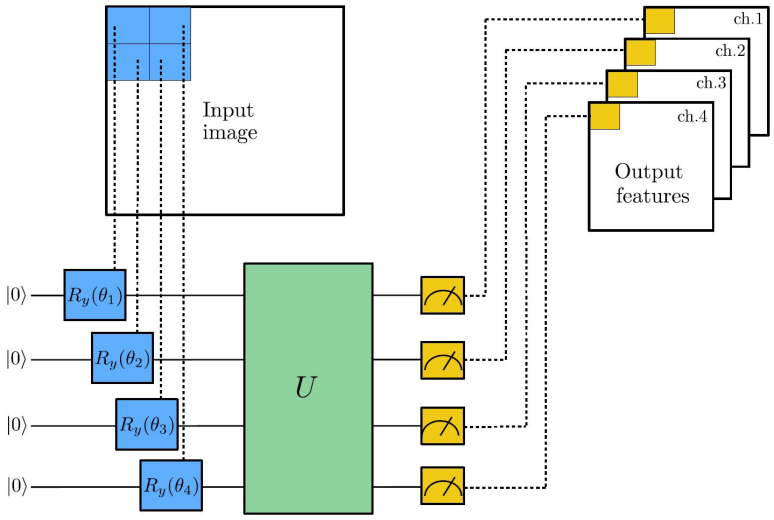
QNN modeline uygulanan Fourier transform işlemi modellerimizin testlerinde kullandığımız 2 farklı piksel değeri olan 3x3 ve 4x4 ile test edilmiştir. 3x3 piksellik resimlere uygulanan Fourier ile yapılan eğitim sonucunda modelimizin doğruluk değeri ~0.72 olarak ölçüldü. Fourier uygulanmayan modele göre doğruluk farkı yaklaşık olarak ~0.11 çıktı.



4x4 piksellik resimlere uygulanan Fourier ile yapılan eğitim sonucunda modelimizin doğruluk değeri ~0.83 olarak ölçüldü. Fourier uygulanmayan modele göre doğruluk farkı yaklaşık olarak ~0.01 çıktı.



3.3) Aşama 3: Quantum Convolutional Neural Network (QCNN) Modeli Oluşturulması



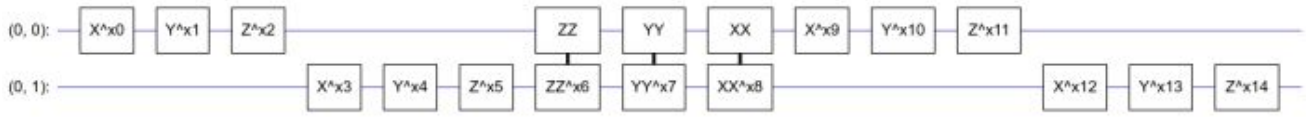
Kuantum CNN modelindeki amacımız ikili resim sınıflandırması yapmaktı. MNIST verisetindeki 3 ve 6 sayılarının sınırlandırılması için baştan sona kuantum algoritması ile çalışan bir kuantum CNN modeli geliştirdik.

Klasik CNN modellerindeki gibi convolutional filtreler oluşturmak için unitary matrisler kullandık. Bu unitary matrisleri X, Y ve Z kapıları olarak parametrized gate halinde tanımladık. Modeli eğitirken amacımız bu kuantum kapılarının öncelikle sembolik parametreler ile başlatılması, ve model eğitilirken parametrelerinin öğrenilmesiydi.

Resmi kuantum modeline girdi olarak vermek için farklı yöntemler denedik. İlk denediğimiz modellerde diğer Kuantum CNN modellerinden farklı olarak resmi çok küçük boyutlara indirmemek için resmi bölme yöntemini kullandık. Öncelikle resimi filtrelerin boyutunda kesip her parça için farklı bir model eğitme ve sonda oylama yöntemi ile resmin sınıflandırmasına karar verme yolunda modeller eğittik. Fakat bu modelin performansı düşük çıktı.

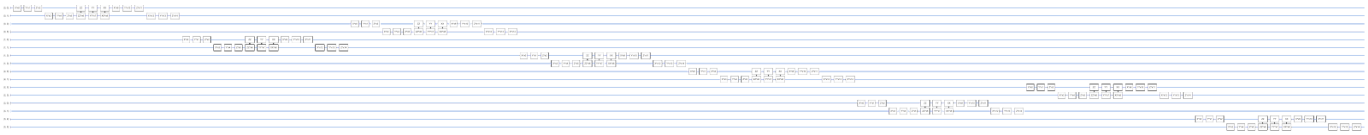
Daha sonra resmin tümünü 6x6 boyutuna indirgeyip ortadaki 4x4'lük parçasını alarak, resmin piksellerini 16 tane kübite dönüştürdüğümüz iki tane model tasarladık.

Bu yöntem ile kurduğumuz birinci modelimizde convolutional filtrelerin boyutlarını 1x2 olarak tanımladık. Buradaki amacımız ikili kübitler üzerinde çalışan kuantum kapılarını kullanabilmek oldu. Oluşturduğumuz filtrelerin ikili kübitler üzerinde devre gösterimi aşağıdaki resimde görülebilir:



İkili Kübit Kapıları

Kurduğumuz kuantum convolutional filtreleri resmin üzerinde art arda gelen kuantum durumuna dönüştürülmüş pikseller üzerine uyguladık. Devrenin tümü aşağıdaki resimde mevcuttur:



Convolution Operasyonu Devresi

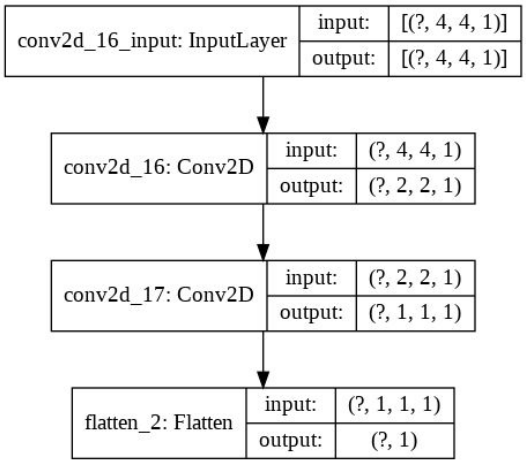
Convolution operasyonundan sonra elde ettiğimiz kuantum durumuna Pauli Z operatörü kullanarak ölçüm yaparak devreyi tamamladık.

İkinci QCNN modelimizi 2x2 filtreli ve 2 katman içeren bir model olarak tasarladık. Birinci modelde kullandığımız ikili kübit kapılarını bağlayarak resmin 2x2'lik kısımlarına uygulanabilecek filtreler elde ettik. Filtreler, her bir 2x2lik kısımdan geçtikten sonra ikili kübit kombinasyonları üzerinde CNOT kapısı uygularak her filtreden tek bir değer çıkardık. Dörtlü kübit kapısı aşağıdaki resimde görülebilir.

w

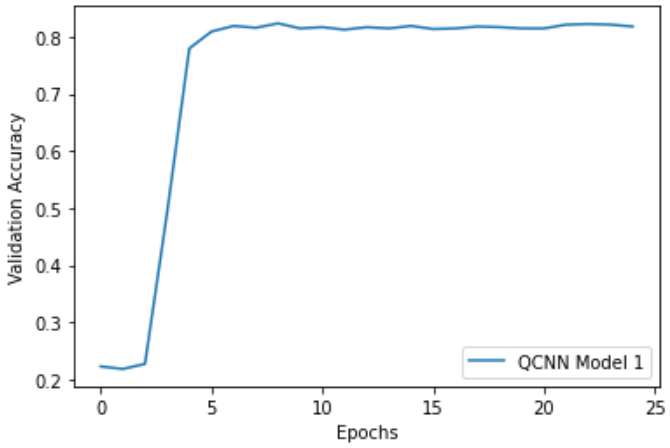
Birinci katmandan geçtikten sonra oluşan 2x2'lik kuantum durumuna tekrar 2x2 boyutunda Convolutional filtre uyguladık. İki katmandan sonra elimize geçen değer üzerinde ölçüm yaparak modeli tamamladık.

QCNN modelimizi klasik CNN modelleri ile karşılaştırmak için kuantum modelimize benzer basit bir CNN modeli geliştirdik. Bu modelde de 2x2 boyutunda filtreler kullanıp iki katman kullandık. QCNN modelinde aktivasyon fonksiyonu kullanmadığımız için bu modelde de kullanmadık. Modelin detayları aşağıdaki grafikte görülebilir:

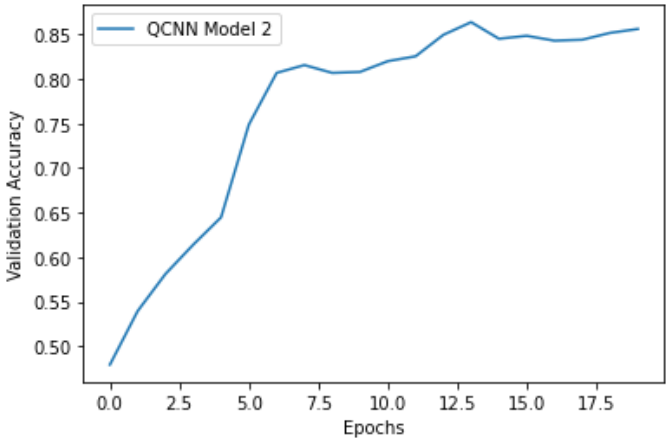


3.3.1) Performans

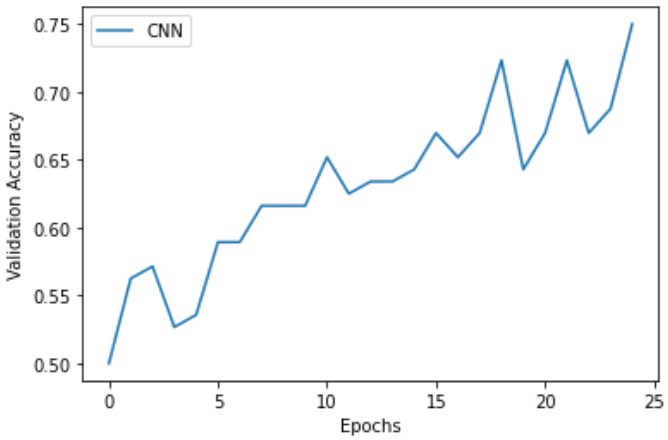
İlk modelimiz MNIST veriseti üzerinde eğitildi. 100 adet resim eğitim, 20 adet resim ise validasyon için kullanıldı. Modelimizin testlerinde ~0.80 doğruluk oranı elde ettik. Eğitim sonucunda ulaştığımız doğruluk değeri ~0.81 oldu. Doğruluk değerinin epoch sayısına göre artışı da aşağıdaki grafikte görülebilir:



İkinci modelimiz de yine MNIST veriseti üzerinde aynı veriler ile eğitildi. Modelin testlerinde ~0.85 doğruluk oranı elde ettik. Doğruluk değerinin epoch sayısına göre artışı da aşağıdaki grafikte görülebilir:



QCNN modelleri ile karşılaştırmak için kurduğumuz klasik CNN modelini de aynı veri seti ile eğittik. Bu modelden ~0.75 doğruluk oranı elde ettik. Doğruluk değerinin epoch sayısına göre artışı da aşağıdaki grafikte görülebilir:



Benzer modeller kurup aynı kayıp fonksiyonları ile ve aynı epoch sayısında eğittiğimiz zaman QCNN modelinin CNN modeline göre daha yüksek doğruluk oranına ulaştığını gördük. Kuantum sistemlerinin gelişmesi ile daha büyük resimlerde ve veri setlerinde resim işleme alanında QCNN modellerinin kullanabileceğini düşünüyoruz.

4) Sonuç

Hackathon'a, kuantum programlama ile ilgili çok az, kuantum makine öğrenmesi ile ilgili ise hiçbir deneyimi olmayan ve öncesinde birbirini tanımayan 4 kişi olarak başladık. 48 saatlik bu maratonda ekip olarak çok iyi bir iletişim kurduk. Bunun sonucunda, Hackathon Discord'dundaki en aktif gruplardan olduk ve haftasonumuzu oldukça eğlenceli geçirdik. Hackathonlarda araştırmalar oldukça yoğun geçtiğinden dolayı kuantum programlamanın birçok alanıyla ilgili güzel bir deneyim edindik. Ve bu deneyimi, bilmediğimiz bir alanda geçirdiğimiz 48 saat içindeki isteğimiz ve gayretimiz ile akademi kategorisinde aldığımız birincilik ile süsledik. Heyecanımızın hiç eksilmediği bu Hackathon'u organize eden QTurkey'e teşekkürlerimizi sunarız ve rekabet ettiğimiz takımları tebrik ederiz.

5) Referanslar

[1]: Implementation and Analysis of Quantum Fourier Transform in Image Processing
[2]: P. Le, F. Dong, and K. Hirota, "A flexible representation of quantum images for polynomial preparation, image compression, and processing operations," Quantum Information Processing, vol. 10, no. 1, pp. 63-84, 2011
[3]: Y. Zhang, K. Lu, Y. Gao, and M. Wang, "NEQR: a novel enhanced quantum representation of digital images," Quantum Information Processing, vol. 12, no. 8, pp. 2833-2860, 2013.
[4]: Performing Quantum Computer Vision Tasks onIBM Quantum Computers and Simulators
[5]: https://pennylane.ai/qml/demos/tutorial_quanvolution.html
[6]: https://www.tensorflow.org/quantum/tutorials/
[7]: Implementation and Analysis of Quantum Fourier Transform in Image Processing, Ola Al-Ta'ani, Ali Mohammad Alqudah, Manal Al-Bzoor
[8]: Quantum neural network, M.V.Altaisky
[9]: Learning the quantum algorithm for state overlap, Lukasz Cincio, Yiğit Subaşı, Andrew T. Sornborger, Patrick J. Coles
[10]: Quantum Natural Gradient, James Stokes, Josh Izaac, Nathan Killoran, Giuseppe Carleo
[11]: Classification with Quantum Neural Networks on Near Term Processors, Edward Farhi, Hartmut Neven
[12]: Image Processing in Quantum Computers, Aditya Dendukuri, Khoa Luu
[13]: Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment, Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, Wenqiang Zheng, Jianzhong Li, Meisheng Zhao, Xinhua Peng, Dieter Suter