Derin Öğrenme Modellerinin Optimizasyonu ve EdgeTPU ile Uyumlu Hale Getirilmesi

Ömer Faruk ERKOÇ

Özet
Son yıllarda teknolojinin gelişimiyle birlikte yapay zeka kavramı hayatımızın her alanına girmiş durumdadır. Yapay zeka alanında kullanılan derin öğrenme teknikleri oldukça başarılı sonuçlar vermesine rağmen çoğunlukla fiyat ve zaman açısından maliyetli tekniklerdir. Bu zaafların giderilmesi, özellikle modellerin çıkarım (infrance) kısmında harcadığı zamanı azaltmak için, eğitim sonrası tam tam sayı kuantalama (Full integer post-training quantization) basitleştirme tekniği uygulanmıştır. Aynı zamanda modelin, derin öğrenme modellerinin işlemci ve grafik kartlarına göre daha hızlı bir şekilde çıkarım yapmasını sağlayan EdgeTPU üzerinde çalıştırılması amaçlanmıştır.

Çalışmada kullanılan derin öğrenme modeli görüntü işleme alanında bir nesne algılama modeli olan EfficientDetLite3'tür. Modelin eğitilmesi için kuş bakışı, 75 tane araba ve 75 tane tır fotoğrafından oluşan bir veri seti hazırlanmıştır. Eğitilen model COCO değerlendirme metriklerine göre değerlendirilmiştir. Modelde yer alan 424 işlemin 273'ünün EdgeTPU üzerinde çalıştırılması başarılmıştır.

1. Giriş

Son yıllarda teknolojinin gelişmesiyle birlikte yapay zeka ve makine öğrenmesi konularında büyük ilerlemeler kaydedilmiştir. Yapay zeka, görevleri yerine getirmek için insan zekasını taklit eden ve topladıkları bilgilere göre yinelemeli olarak kendilerini iyileştirebilen sistemler veya makineler anlamına gelir[1]. Makine öğrenimi (ML), tükettikleri verilere göre öğrenen ya da performansı iyileştiren sistemler oluşturmaya odaklanan bir yapay zeka (AI) alt kümesidir[2]. Yapay sinir ağları (YSA), insan beyninin bilgi işleme tekniğinden esinlenerek geliştirilmiş bir bilgi işlem teknolojisidir. YSA aslında biyolojik nöron hücrelerinin ve bu hücrelerin birbirleri ile arasında kurduğu sinaptik bağın dijital olarak modellenmesidir[3]. Derin öğrenme (aynı zamanda derin yapılandırılmış öğrenme, hiyerarşik öğrenme ya da derin makine öğrenmesi) bir veya daha fazla gizli katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Yani en az bir adet yapay sinir ağının (YSA) kullanıldığı ve birçok algoritma ile, bilgisayarın eldeki verilerden yeni veriler elde etmesidir[4]. Derin öğrenme modelleri çoğunlukla yapay sinir ağları içerdiği için bu modellere derin sinir ağları da (deep neural networks) denir.

Derin öğrenme modellerinin son yıllarda popülerlik kazanmasının temelde iki nedeni vardır, bunlardan ilki teknolojinin gelişmesiyle birlikte donanım anlamındaki limitlerin ortadan kalkmış olmasıdır. Bu modellerin eğitilmesinde kullanılan grafik işlem birimi (GPU), bellek (RAM), merkezi işlem birimi (CPU) gibi donanımların geçmişe kıyasla çok daha güçlü olmaları sayesinde derin öğrenme modellerinin geliştirilmesi ve çalıştırılması çok daha uygulanabilir hale gelmiştir. İkinci neden ise bu modellerin çok büyük başarı göstermesi ve bir çok uygulama alanına sahip olmasıdır.

Bu modeller her ne kadar görüntü işleme ve doğal dil işleme gibi bir çok alanda çok başarılı sonuçlar ortaya koyuyor olsalar da bu başarının arkasındaki en temel sebeplerden biri

çok sayıda katmandan oluşmaları ve milyonlarca hatta milyarlarca parametre içermeleridir. Bu nedenle bu modelleri eğitmek ve çalıştırmak için güçlü sistemler ve yüksek depolama alanları gerekmektedir. Dolayısıyla zaman, işlem gücü ve para açısından oldukça maliyetli modellerdir.

Örnek olarak OpenAI tarafından geliştirilen GPT-3 modeli 175 milyar parametre[6] içermektedir ve modelin eğitilmesinin maliyeti 4.6 milyon dolardır[7]. Bu projede kullanılan EfficientDet-Lite3 modeli ise 8.4 milyon parametre içermektedir[8]. Aynı zamanda bu kadar çok parametreye sahip olan modeller, hem yüksek depolama alanına ihtiyaç duymaları hem de kısa sürede çıkarım yapabilmek için yüksek işlem gücüne ihtiyaç duymaları nedeniyle, özellikle donanım anlamında daha kısıtlı olan uç (edge) cihazlarda çalıştırılmaya uygun değillerdir. Bu nedenle bu modelleri uç cihazlarda çalıştırabilmek için budama (pruning), kuantalama (quantization), bilgi damıtma (knowledge distillation), düşük dereceli çarpanlara ayırma (low-rank factorization) gibi basitleştirme teknikleri uygulanır. Bu teknikler uygulanırken, modelin boyutunu küçültmek veya çıkarım yapma süresini kısaltmak amaçlanır, aynı zamanda modelin çıkarım yapma konusundaki performansını azaltmamak gözetilir.

Bu çalışmada EfficientDetLite3 modeline eğitim sonrası tam sayı kuantalama tekniği uygulamak ve EdgeTPU üzerinde çalışabilir kılmak amaçlanmıştır.

2. İlgili Araştırma

2.1 Basitleştirme Teknikleri

Modelleri basitleştirme fikri, derin öğrenme modellerindeki bir çok parametrenin modelin çıkarım yapma performansına katkıda bulunmadığının gözlemlenmesi ile ortaya çıkmıştır. Basitleştirmede ki amaç, orijinal modelin çıkarım performasına yakın bir performansa sahip olan ancak daha az işlem gücüne ihtiyaç duyacak veya çıkarım yapma süresi daha hızlı olacak bir model elde etmektir[9].

Budama (pruning) tekniği, model performansını etkilemeyecek ya da minimum oranda etkileyecek olan gereksiz parametre, bağlantı ya da katmanların silinmesi esasına dayanan bir tekniktir. Budama sayesinde modelin çıkarım yapma performansı en fazla kabul edilebilir bir seviyede azalırken, modelin hem bellekte kapladığı alan hem de çıkarım yapması için gereken işlem gücü azaltılmış olur. Gereksiz olan parametre, bağlantı ya da katmanlar, belirli bir eşik seviyesi (threshold) uygulanarak ya da rastgele seçilerek modelden silinir.

Kuantalama (quantization) tekniği, derin öğrenme modellerinde normalde 32-bit kayan noktalı sayılar (floating-point numbers) olarak depolanan ağırlıkları ve aktivasyonları, 16-bit, 8-bit, 4-bit hatta 1-bit olarak depolayabilmek için uygulanan bir tekniktir[10]. Kullanılan bit sayısının azaltılması, modelin gerektirdiği depolama alanın da azalmasını sağlar. Kuantalama, eğitim sonrası kuantalama (post-training quantization) ve kuantalama farkında eğitim (quantization aware training) olmak üzere iki farklı şekilde uygulanabilir. Eğitim sonrası kuantalama, model eğitildikten sonra uygulanır bu nedenle model üzerinde bir

modifikasyon yapmayı gerektirmez. Eğitim sonrası tam tam sayı kuantalama tekniğinde, bellekte 32 bit yer kaplayan ve kayan noktalı sayılar(float) olarak tutulan ağırlık ve aktivasyonlar bellekte 8 bit yer kaplayan tam sayı (integer) veri tipine dönüştürülür. Bu yöntemde modelin kalibre edilebilmesi için asıl veri setinden oluşturulacak olan daha küçük ve temsilci olarak adlandırılan bir veri seti ile model tekrar eğitilmelidir[11]. Kuantalama farkında eğitim, modele kuantalama sonucunda ortaya çıkacak olan performans düşüklüğünü simüle edecek olan sahte düğümler (node) eklenerek gerçekleştirilir. Bu nedenle genel olarak eğitim sonrası kuantalamaya göre daha başarılı modeller elde edilir.

Düşük dereceli çarpanlara ayırma tekniği, mümkün olduğu kadar fazla bilgiyi korurken onları daha verimli bir şekilde depolamak için bir matrisi daha düşük dereceli matrislere dönüştürmeyi içeren bir tekniktir. Yoğun katmanlar için kullanıldığında modeli depolama alanı açısından iyileştirirken konvolüsyon katmanları için kullanıldığında modelin çıkarım yapma hızını iyileştirir[10].

Bilgi damıtımı (knowledge distillation) tekniği, bilgiyi aktararak küçük modelleri daha büyük modeller kadar doğru olacak şekilde eğiten bir eğitim tekniğidir. Bilgi damıtma alanında, daha büyük modele "öğretmen ağı" denirken, daha küçük ağ "öğrenci ağı" olarak bilinir. Öğretmen modeli eğitildikten sonra, damıtma işlemi, eğitilen bilgiyi yakalayıp daha küçük öğrenci modeline aktarabilir ve ardından öğrenci ağını eğitebilir. Tepki tabanlı bilgi damıtması(response-based knowledge distillation), öğretmen ağının çıktı katmanından (çıkarımlar) bilgileri yakalar ve aktarır. Öğrenci ağı, damıtma kaybını en aza indirerek bu son tahminleri doğrudan taklit eder. Özelliğe dayalı bilgi ayrıştırma (feature-based knowledge distillation), öğretmen ağının ara ve çıktı katmanlarının bilgilerini yakalar. İlişki tabanlı bilgi damıtması(Relation-based knowledge distillation), öğretmen ağının ara ve çıktı katmanlarının ötesine geçer. Farklı veri örnekleri ve katmanları arasındaki ilişkileri araştırır[12].

Song Hang, Huizi Mao ve William J. Dally'nin yaptığı çalışmada[13] yukarıdaki basitleştirme tekniklerinden budama ve kuantalama kullanılarak ve bunlara ek olarak da huffman coding kullanılarak VGG-16 modelinin çıkarım yapma performansında bir kayıp olmadan modelin boyutunu 552MB'dan 11.3MB'a düşürmeyi başarmışlardır.

2.2 EdgeTPU

EdgeTPU, Düşük güce sahip olan cihazlarda yüksek performanslı model çıkarımları yapmak için Google tarafından geliştirilen uygulamaya özgü tümleşik devredir (application specific integrated circuit). Derin öğrenme modellerini eğitmekten çok bu modellerin düşük performanslı cihazlarda çıkarım yapma sürecini enerji verimliliği ve hız açısından iyileştirir[16].

2.3 Öğrenme Aktarımı (Transfer Learning)

Öğrenme aktarımı, önceden eğitilmiş bir modeli yeni bir görevdeki model için başlangıç noktası olarak yeniden kullandığımız bir makine öğrenimi yöntemidir. Bir görev üzerinde eğitilmiş bir model, ikinci görevi modellerken hızlı ilerlemeye izin veren bir optimizasyon olarak ilgili ikinci bir görev için yeniden tasarlanır.[14]. Diğer bir deyişle,

önceden eğitilmiş bir modeldeki ağırlıklar üzerinde değişiklikler yapılarak yeni bir görev için kullanılabilir. Öğrenme aktarımı sayesinde, küçük veri setleri ile yüksek çıkarım performansına sahip olan modeller elde edilebilir.

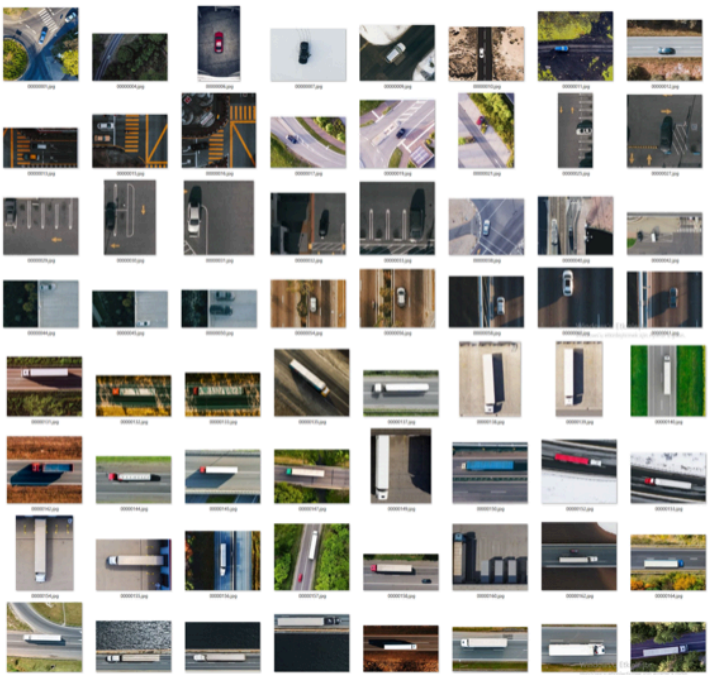
2.4 TensorflowLite

TensorFlow Lite, geliştiricilerin modellerini mobil, gömülü ve uç cihazlarda çalıştırmasına yardımcı olarak cihaz üzerinde makine öğrenimini etkinleştiren bir dizi araçtır[15].TensorflowLite ile düşük donanıma sahip olan cihazlarda çalışacak modellerin geliştirilmesi kolaylaştırılmıştır. TensorflowLite ile bir nesne tanıma modeli geliştirmek için TFLite Model Maker ve ObjectDetector API'si kullanılmalıdır. Bu API sayesinde öğreme aktarımı kullanılarak bir nesne tanıma modelini özel bir veri seti için eğitilebilir. ObjectDetector API'si, modele verilecek olan veri setini kullanmak için DataLoader modülünü kullanmayı zorunlu kılar.

3. İlgili Çalışmalar ve Sonuç

3.1 Veri Seti

Bu projede 75 tane araba ve 75 tane tır fotoğrafından oluşan bir veri seti kullanılmıştır. Veri setindeki fotoğraflar önce jpg formatına dönüştürülüp, sonrasında makesense.ai isimli web sitesi üzerinden Pascal VOC XML formatında etiketlenmiştir (annotate). Veri seti eğitim, test ve doğrulama (validation) olmak üzere üç alt sete bölünmüştür. Eğitim seti 110, test ve doğrulama setleri 20'şer fotoğraf içermektedir.



3.2 Model Yapısı

Nesne tanıma için kullanılan model EfficientDetLite3 modelidir. Bu model TensorflowLite kütüphanesinde yer alan Model Maker'ın Object Detector API'si kullanılarak transfer learning ile veri seti üzerinde 50 epoch eğitilmiştir. Modelin yapısı:

```
model = object_detector.create(train_data=train_data,
                               model_spec=object_detector.EfficientDetLite3Spec(),
                               validation_data=validation_data,
                               epochs=50,
                               batch_size=10,
                               train_whole_model=True)
```

3.3 Sonuçlar

Oluşturulan modelin COCO değerlendirme metriklerine göre sonuçları:

```
model.evaluate(test_data)

1/1 [==============================] - 7s 7s/step

{'AP': 0.82409716,
 'AP50': 0.97331446,
 'AP75': 0.9351219,
 'APs': -1.0,
 'APm': 0.7818864,
 'APl': 0.8523784,
 'ARmax1': 0.70357144,
 'ARmax10': 0.885119,
 'ARmax100': 0.88928574,
 'ARs': -1.0,
 'ARm': 0.8736111,
 'ARl': 0.90125,
 'AP_/car': 0.8852399,
 'AP_/truck': 0.76295435}
```

Daha sonra model, Tensorflow Lite formatına dönüştürülür. Bu dönüşüm yapılırken modele otomatik olarak eğitim sonrası tam tam sayı kuantalama uygulanır. Temsilci veri seti olarak, modeli eğitirken kullandığımız veri seti kullanılır.

Tensorflow Lite formatına dönüştürürken eğitim sonrası tam tam sayı nicemleme tekniği kullanıldığı için modelin çıkarım yapma performansında çeşitli değişiklikler meydana gelecektir. Bu nedenle Tensorflow Lite formatındaki model yeniden değerlendirilmelidir. Tensorflow Lite formatındaki modelin COCO değerlendirme metriklerine göre sonuçları:

```
model.evaluate_tflite(TFLITE_FILENAME, test_data)

20/20 [==============================] - 371s 19s/step

{'AP': 0.8054606,
 'AP50': 0.96869403,
 'AP75': 0.92914444,
 'APs': -1.0,
 'APm': 0.7532178,
 'APl': 0.83869636,
 'ARmax1': 0.6964286,
 'ARmax10': 0.8428571,
 'ARmax100': 0.8428571,
 'ARs': -1.0,
 'ARm': 0.79444444,
 'ARl': 0.865,
 'AP_/car': 0.85501766,
 'AP_/truck': 0.7559034}
```

Modelin yaptığı çıkarımlardan örnekler:

1-

2-

Model’in EdgeTPU için derlenmesi:

```
Model successfully compiled but not all operations are supported by the Edge TPU. A percentage of the model will instead run on the CPU,
Number of operations that will run on Edge TPU: 273
Number of operations that will run on CPU: 151

Operator                       Count      Status

QUANTIZE                       1          Mapped to Edge TPU
CONV_2D                        58         More than one subgraph is not supported
CONV_2D                        94         Mapped to Edge TPU
CUSTOM                         1          Operation is working on an unsupported data type
ADD                            72         Mapped to Edge TPU
ADD                            11         More than one subgraph is not supported
RESHAPE                        10         More than one subgraph is not supported
MAX_POOL_2D                    4          More than one subgraph is not supported
MAX_POOL_2D                    22         Mapped to Edge TPU
CONCATENATION                  2          More than one subgraph is not supported
LOGISTIC                       1          Operation is otherwise supported, but not mapped due to some unspecified limitation
DEPTHWISE_CONV_2D              64         Mapped to Edge TPU
DEPTHWISE_CONV_2D              58         More than one subgraph is not supported
RESIZE_NEAREST_NEIGHBOR        20         Mapped to Edge TPU
RESIZE_NEAREST_NEIGHBOR        4          More than one subgraph is not supported
DEQUANTIZE                     2          Operation is working on an unsupported data type
Compilation child process completed within timeout period.
Compilation succeeded!
```

Görüleceği üzere modelde yer alan işlemlerden 273 tanesi EdgeTPU ile uyumlu hale getirilmiştir. Geri kalan 152 işlem CPU üzerinde çalışmaya devam edecektir.

Kaynakça:

[1] Oracle, “Yapay Zeka (AI) nedir?”, erişim 10 Ekim 2022, https://www.oracle.com/tr/artificial-intelligence/what-is-ai/

[2] Oracle, “ Makine Öğrenimi nedir?”, erişim 12 Ekim 2022 https://www.oracle.com/tr/artificial-intelligence/machine-learning/what-is-machine-learning/

[3] Wikipedia, “Yapay Sinir Ağları”, erişim 12 Ekim 2022 https://tr.wikipedia.org/wiki/Yapay_sinir_a%C4%9Flar%C4%B1

[4] Wikipedia, “Derin Öğrenme”, erişim 12 Ekim 2022 https://tr.wikipedia.org/wiki/Derin_%C3%B6%C4%9Frenme

[5] Simonyan, K. ve Zisserman A. (2015), “Very Deep Convolutional Networks For Large-Scale Image Recognition”, ICLR

[6] Tan, M. Pang, R *et al.* (2020), “EfficientDet: Scalable and Efficient Object Detection”, Google Research Team

[7] heits.digital, “The Hitchhiker’s Guide to GPT3”, erişim 15 Ekim 2022, https://heits.digital/articles/gpt3-overview

[8] Github, “automl/efficientdet”, erişim 15 Ekim 2022, https://github.com/google/automl/tree/master/efficientdet

[9] Medium, “An Overview of Model Compression Techniques for Deep Learning in Space”, erişim 20 Ekim 2022, https://medium.com/gsi-technology/an-overview-of-model-compression-techniques-for-deep-learning-in-space-3fd8d4ce84e5

[10] Xailient, “4 Popular Model Compression Techniques Explained”, Erişim 21 Ekim 2022, https://xailient.com/blog/4-popular-model-compression-techniques-explained/

[11] Tensorflow, “Post-training quantization”, erişim 15 Kasım 2022, https://xailient.com/blog/4-popular-model-compression-techniques-explained/

[12] deci.ai, “Introduction to Knowledge Distillation”, erişim 20 Kasım 2022, https://deci.ai/blog/knowledge-distillation-introduction/

[13] Han, S. ve Dally, W. J. *et al.* (2016), “Deep Compression: Compressing Deep Neural Networks With Pruning, Trained Quantization and Huffman Coding”, ICLR

[14] V7labs, “What is Transfer Learning”, erişim 25 Kasım 2022, https://www.v7labs.com/blog/transfer-learning-guide

[15] Tensorflow, “TensorFlow Lite”, erişim 27 Kasım 2022, https://www.tensorflow.org/lite/guide

[16] Coral.ai, “What is the Edge TPU?”, Erişim 5 Kasım 2022, https://coral.ai/docs/edgetpu/faq/