FİGES

12/12/2023

Deep Learning
Onramp

Doğukan Mehmet
KILINÇ

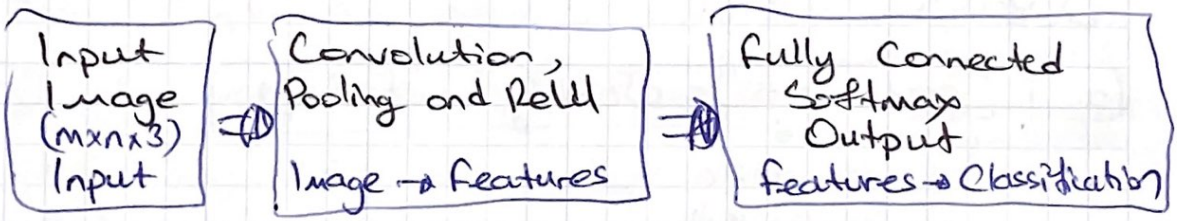
① `imread()` → Görseli okur,
② `imshow()` → Görseli ekrana yazdırır.

Önceden eğitilmiş, tanımlanmış GoogleNet mimarisini kullanabilmek için

③ `net = googlenet;` atamasını yapabiliriz.

Bir görüntü üzerinde sınıflandırma yapabilmek için "classify" fonksiyonunu kullanabiliriz.

④ `pred = classify(net, img)`



GoogleNet atanması yaptığımız ③'teki katmanları çekmek için aşağıdaki gibi nokta gösterimi kullanırız.

③' kod → `net = googlenet;` idi.

⑤ `ly = net.Layers` kullanarak katmanı çekebiliriz

ly dizisi bir ağ katmanı dizisidir. ly üzerinde indeksleme kullanarak tek bir katman incelenebilir.

⑥ `layer3 = ly(3)`

Her ağ katmanı, o katman türüyle ilgili özelliklere sahiptir. Giriş katmanının önemli olan özelliği, ağın giriş olarak beklediği görüntülerin boyutları olan "Input Size"dır. Nokta gösterimi ile özellik değiştirilir.

Gri tonlamalı görüntüler MATLAB'de m'ye n diziı olarak (m x n) ifade edilir. Her öğenin değeri, karşılık gelen görüntü pikselinin yoğunluğunu temsil eder.
Renkli görüntüler m'ye n'ye 3'lük bir dizi olarak (m x n x 3) temsil edilir.

Bir çıktı katmanının "Classes" özelliği, ağın tahmin etmek üzere eğitildiği kategorilerin adını verir.
Bu 1000 resim 224 x 224 x 3 boyutundadır.
GOOGLENET INPUT SIZE

"classify" işlevi, ağın en yüksek puanı atadığı çıktıyı ekrana verir. İkinci bir çıktı istediğiniz sınıflar için tahmin edilen puanları alabilirsiniz.

(7) `[pred, scrs] = classify(net, img)`

Tüm skorları görmek için

Ardından bu elde edilen skorları bar grafik yardımı ile görselleştirebiliriz. Bazen çok fazla kategori olmuş olabilir. Bu konuda da grafiğin bir bölümü üzerine sınırlama yapabiliriz.

(8) `(bar (scrs)`

Sınırlama yapmak, daha net bir grafik ortaya koymak için eşik değerleri kullanabiliriz.

(9) `highscores = scores > 0.01 :`
(scrs)

Ardından scrs içerisinden highscores'i çekip bunu bar grafiğine yansıtırız.

(10) `bar(score(highscore))`

Bu işlemler sonucu x-ekseninin isimlerini ilgili değerler için atamak istersek aşağıdaki komutu kullanırız

(11) `xticklabels(categorynames(highscores))`

kategori isimleri arasında belirli eşik değerlere ait olanları yaz

Bu eşik değere ait olan kategori isimlerini x-ekseninde ilgili yere yazar

DATASTORE

MATLAB'da bir veri deposu oluşturmak için, klasör veya dosya adımlarını girdi olarak belirterek "imageDatastore" işlevini kullanabiliriz. Birden fazla dosya belirtmek için (*) gibi joker karakterler kullanılabilir.

(12) `ds= imageDatastore("foo*.png")` → foo1, foo2 ortak tanımlayıcı ad yazılır.

↳ Bu dosya PNG dosyaları için foo ile başlayan adlara sahip bir veri deposu oluşturacaktır.

"read", "readimage" ve "readall" işlevi kullanarak verileri bir veri deposunda manuel olarak içe aktarabiliriz.

- read: Görüntüleri sırayla birer birer içe aktarır.
- readimage: Tek bir spesifik görüntüyü içe aktarır.
- readall: Tüm görüntüleri tek bir hücre dizine aktarır. Her görüntü ayrı ayrı hücrededir.

(13) `I= readimage(ds,n)`

↳ Bu dsiteki n. görüntüyü I'ya aktarır.

Sınıflandırma için CNN işlemlerinde tek bir görüntü yerine bir görüntü veri deposu kullanabiliriz.

(14) `preds = classify(net, ds)`

Sonuç her bir görüntü için (veri deposundaki) bir tane olmak üzere, tahmin edilen sınıfların bir dizisi olacaktır.

Görselleri Kullanıma Hazırlama

Giriş Görüntüleri Ayarlama

Daha önceden önceden işlenmiş fotoğraflar kullandık Burada görselleri GoogleNet ile sınıflandırılacağı için yeniden boyutlandırılmış olacaktır. Başlangıçta farklı boyutta olabilirler fakat ağı eğitmek için ağa uygun boyutta olmalı.

Bir ağın giriş katmanı, ağın ihtiyaç duyduğu görüntü boyutunu verir.

(15) `expectedSize = inputLayer.InputSize`

(UZUN HALİ)

```
net = googlenet;                    % net'i yükler
(16) inlayer = net.Layers(1);       % ilk katmanı alır.
insize = inlayer.InputSize          % input size özelliğini getirir.
```

Bir görüntüyü beklenen giriş boyutuyla eşleşecek şekilde yeniden boyutlandırmak için "imresize" kullanılır.

(17) `imrest = imresize(img, [numrows numcols]);`

Bu img'yi verilen boyutlara göre yeniden biçimlendirir. Artık görsel GoogleNet'in giriş katmanının beklenen boyutlarıyla eşleştiğine göre görsel sınıflandırılabilir.

FİGES

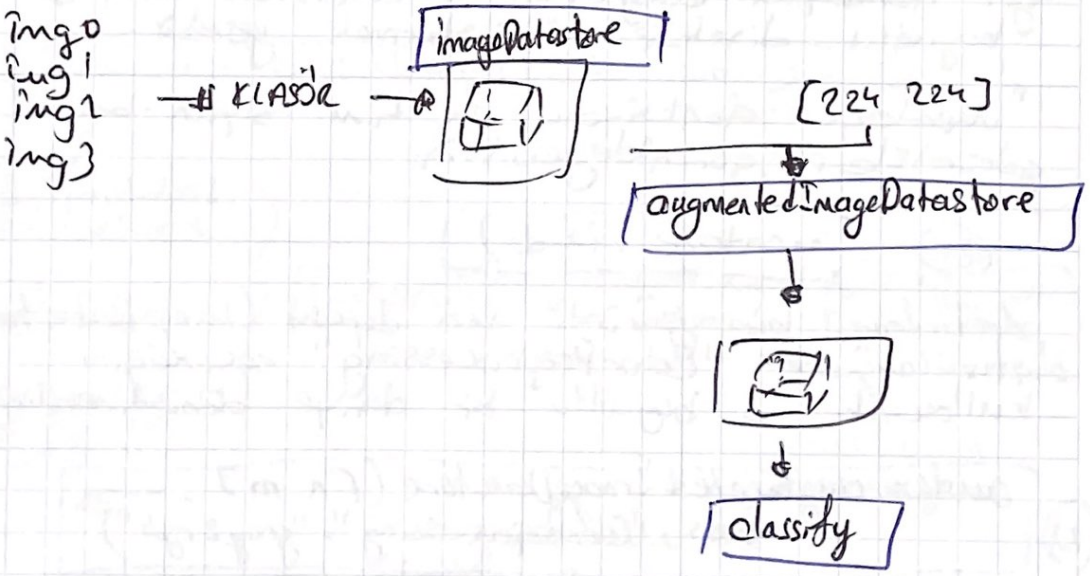
...../...../20.....

GÖRÜNTÜ ÖNİŞLEME İŞ AKIŞI

Görüntüleri evrişimli bir sinir ağıyla sınıflandırmak için, her görüntünün ağının giriş katmanı tarafından belirtilen boyutlara sahip olması gerekir. Görüntülere genellikle sınıflandırılmadan önce basit bir ön işleme gerekir.

Her görüntü ayrı ayrı da işlenebilir fakat bu zaman alabilir. Bunun yerine imageDatastore (görüntü veri deposu) kullanılır. Bunun için tüm verilerin bir klasöre kaydetmek gerekir.

Görüntülerin evrişimli bir sinir ağı ile sınıflandırılması için her görüntünün ağın giriş katmanındaki belirtilen boyuta sahip olması gerekir. Görüntüleri tek tek sınıflandırmak yerine bir klasörden imageDatastore formatına, oradan da boyutlandırma işlevine ilerlemek daha faydalıdır.



FİGES

...../...../20.....

Veri Deposundaki (imageDatastore) Görüntüleri Yeniden Boyutlandırma

Tüm imageDatastore formundaki veriye basit ön işleme adımları gerçekleştirilebilir.

(18) `auds = augmentedImageDatastore([r c], imds)` ← eğitim imageDatastore

rows: r, columns: c

Ardından bunu sınıflandırma aşamasına sokabilirsiniz.

(19) `preds = classify(net, auds)`

RGB Görüntüleme Dönüştürme

Gri tonlamalı görüntüler ($n \times m \times 1$) boyutundadır. GoogleNet gibi bir çok önceden eğitilmiş ağ ($p \times p \times 3$) boyutunda görüntü gerektirir. Resim gri tondaysa GoogleNet ile kullanmak için 3 boyutlu dizelere dönüştürmek gerekir.

"montage" fonksiyonu ile tüm siyah-beyaz görüntüleri görüntüleyebiliriz.

(20) `montage(imds)`

Artırılmış bir görüntü veri deposu (imageDatastore) oluşturulduğunda "ColorPreprocessing" seçeneğini kullanarak 3 boyutlu bir diziye dönüştürebiliriz.

(21) `auds = augmentedImageDatastore([n m], imds, "ColorPreprocessing", "gray2rgb")`

Bu fonksiyon üç boyutlu bir dizi oluşturmak için gri tonlamalı bir görüntüyü üç kez kopyalayacaktır. "gray2rgb" gri tonlamalı bir görüntüyü temsil ederse, işlenen görüntü renkli (RGB) bir görüntüyü temsil eden 3 boyutlu bir dizi olacaktır.

ardından "classify" ile sınıflandırma yapabiliriz

(22) `preds = classify(net, auds)`

imageDatastore fonksiyonu varsayılan olarak verilen klasördeki görüntü dosyalarını arar. Verilen klasörün alt klasöründeki görselleri aramak için "IncludeSubfolders" fonksiyonu kullanılabilir.

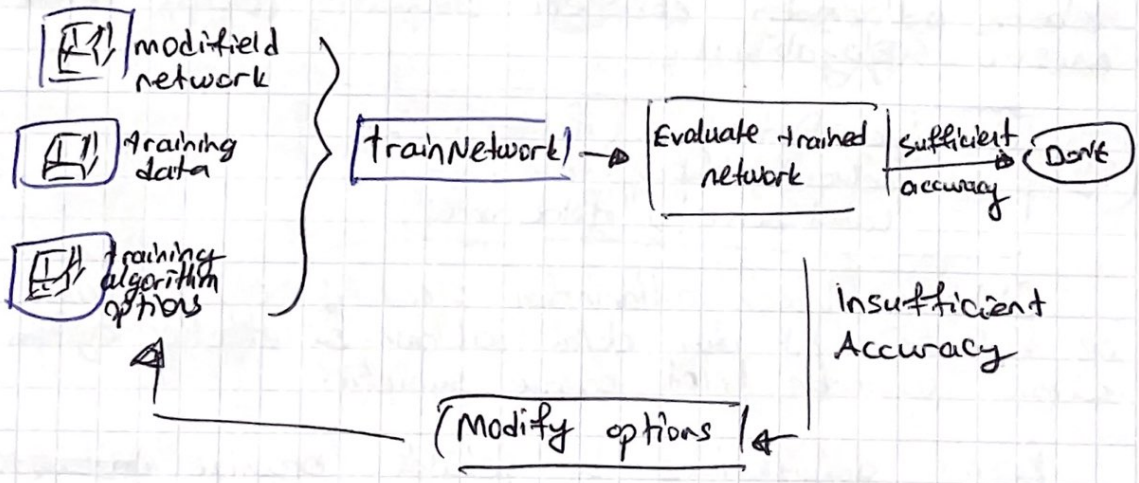
(23)
```
ds = imageDatastore("folder", ...
    "IncludeSubfolders", true)
```

Bir Network'ü eğitmemiz için 3 şeye ihtiyacımız vardır.

1) layers
2) data
3) options

Bunlara sahipsek sistemimizi kolayca aşağıdaki gibi eğitebiliriz.

(24) `newnet = trainNetwork(data, layers, options)`



Transfer öğrenimini gerçekleştirmek için üç bileşen oluşturmamız gerekir:

→ Ağ mimarisini temsil eden bir dizi katman. Transfer öğrenimi için bu, GoogleNet gibi önceden var olan bir ağ değiştirilerek oluşturulur. (LABELS)
→ Eğitim verileri olarak kullanılacak bilinen etiketlere sahip görseller. Bu genellikle bir veri deposu olarak sağlanır. (TRAIN DATA)
→ Eğitim algoritmasının davranışını kontrol eden seçenekleri içeren bir değişken (TRAIN ALGORITHM)

Bu üç bileşen, eğitilmiş ağı çıktı olarak döndüren "trainNetwork" fonksiyonuna girdi olarak sağlanır.
Yeni eğitilen ağın performansını test etmelisiniz. Yeterli değilse, genellikle bazı eğitim seçeneklerini ayarlamayı ve yeniden eğitimi denemelisiniz.

Bir klasör altında saklanan birden fazla klasör içinde eğitim verileri olabilir. Aynı klasördeki veriler aynı özelliklere sahiptir.
"LabelSource" seçeneğini belirleyerek veri deposunun klasör adlarından etiketleri otomatik olarak belirlenmesini sağlayabiliriz.

(25)
```
ds= imageDatastore (folder, ...
    "IncludeSubfolders", true, ...
    "LabelSource", "foldernames")
```

Etiketler, klasör adlarından otomatik olarak okunur ve kategorik bir dizi olarak saklanır. Bu etiketler eğitim süresi açısından kritik öneme sahiptir.

Resim dosyalarınız bu şekilde organize ~~edilmişse~~ edilmemişse, labels özelliğini, kategorik bir resim etiketleri dizisi olacak şekilde manuel olarak değiştirmemiz gerekecek.

Bir veri deposundaki görüntüleri iki ayrı veri deposuna bölmek için "splitEachLabel" işlevini kullanabilirsiniz.

(26) `[ds1, ds2] = splitEachLabel(imds, p)`

p oranı (0'dan 1'e kadar bir değer), imds'deki her etiketin ds1'de yer alması gereken görüntülerin oranını gösterir. Kalan dosyalar ds2'ye atanır.

Varsayılan olarak "splitEachLabel" dosyaları dizeli tutar. İsteğe bağlı "randomized" ekleyerek dosyaları rastgele seçim yapması için talimat verebilirsiniz.

(27) `[ds1, ds2] = splitEachLabel(imds, p, 'randomized')`

Dengesiz Eğitim Verileri

Bazı uygulamalarda bir sınıfta diğerine göre çok daha fazla görüntünün bulunması yaygındır. Örneğin kusurları elde etmeye çalışırken genellikle kabaca çok sayıda görüntü elde etmek kolaydır, ancak kusurlu görüntüleri elde etmek daha zordur.

Eğitim verileri kusursuz görsellerden oluşursa bu durumda kusurlu verilerden yana bir öğrenmeme söz konusu olabilir.

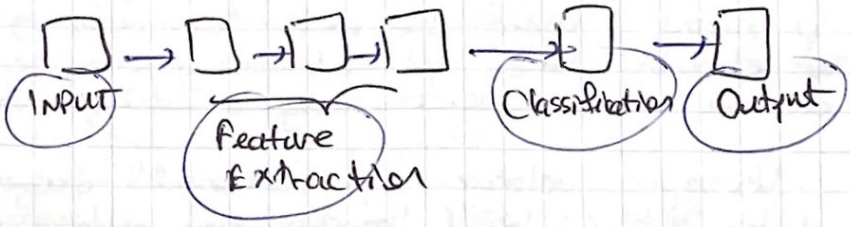
Bunu önlemek için, eğitim görüntülerinin her sınıftan eşit sayıda olmasını sağlayacak şekilde verileri bölmek yararlı olabilir.

"p", 0'dan 1'e kadar bir değer olduğunda oranlı olarak algılanır, yorumlanır. Daha sonra görüntüler, her etiketin oranlı olarak bölünmesini sağlayacak şekilde bölünür. Ayrıca ds1'e atanmak üzere her etiketten alınacak tam sayıda dosya belirtebilirsiniz.

(28) `[ds1, ds2] = splitEachLabel(imds, n)`

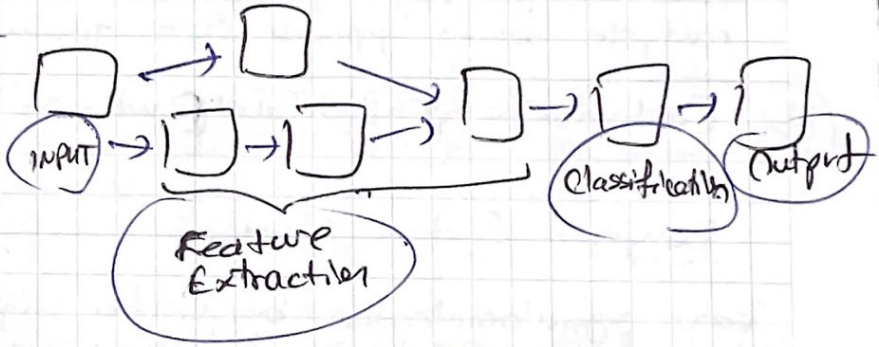
Bu, kategorilerin tümü aynı sayıda görsel içermese bile dsl'deki her etiketin n-görsele sahip olmasını sağlar

SERIES NETWORK



DIRECTED ACYCLIC GRAPH NETWORK (DAG NETWORK)

DAG Network daha fazla kompleks mimariye sahiptir.



SERIES NETWORKS

└ LAYERS

DIRECTED ACYCLIC GRAPH NETWORK (DAG NETWORK)

└ LAYERS
└ CONNECTIONS

"trainingOptions" fonksiyonu kullanarak seçilen bir eğitim algoritması için mevcut seçenekleri görebiliriz

(29) `opts = trainingOptions ("sgdm")`

Çoğu zaman ilk önce seçeneklerin çoğunu varsayılan değerlerde bırakarak antrenman yapmayı denemek isteriz. Fakat en iyi sistemi elde etmek amacıyla çoğu zaman parametrelerle oynarız. Transfer öğrenimi amacı da mevcut bir ağda ince ayar yapmaktır. Bu nedenle genellikle ağırlıkları sıfırdan eğitime göre daha az agresif değiştirmek isteyebiliriz.

"trainingOptions" fonksiyonunda istediğimiz parametreyi istediğimiz şekilde değiştirebiliriz.

(30) `opts = trainingOptions ("sgdm", "Name", value)`

"Name": Değiştirmek istenen parametre adı

value: Parametre değeri

Transfer Learning için 4 şeye ihtiyaç vardır.

- ☑ Network Layers
- ☑ Training Data
- ☑ Algoritham Options

(31) `newnet = trainNetwork (data, layers, options)`

↳ Bu kod ile modelimizi eğitebiliriz. Ardından karşımıza bir tablo çıkacak.

Epoch	Iteration	Time Elapsed (seconds)	Mini Batch Loss	Mini Batch Accuracy	Base Learning Rate

"Mini-Batch" Nedir?

Her yinelemede ağırlıkları güncellemek için "mini-batch" olarak bilinen eğitim görüntülerinin bir alt kümesi kullanılır. Her yinelemede farklı bir "mini-batch" kullanılır. Eğitim setinin tamamında kullanıldığında buna "epoch" denir.

Maksimum dönem sayısı (MaxEpoch) ve mini grupların sayısı (MiniBatchSize), eğitim algoritmaları seçenekleri̇nde ayarlayabileceğimiz parametrelerdedir.

trainingNetwork içerisine "plots", "training-progress" yazacak train işlemi sürecinde train grafiğini bize gösterecektir.

Eğitimden Sonra Ağı Değerlendirme

Aşağıdaki info değişkeni eğitime ilişkin bilgileri içeren bir yapıdadır. TrainingLoss ve TrainingAccuracy alanları, her iterasyondaki eğitim verileri üzerinde oluşan ağın performansının bir kaydını içerir.

INFO

- TrainingLoss
- TrainingAccuracy
- BaseLearnRate

(32) plot(info.TrainingLoss) fonksiyonu ile kayıp grafiğini çizdirebiliriz.

İki dizinin eşlenen öğelerinin sayısını belirlemek için matiksal operatörleri ve nnz'yi kullanabilirsiniz.

(33) `numequal = nnz(a==b)`

↳ Bunu tahmin edilen test sonuçları ile gerçek test sonuçları sonucu doğruluğun yüzde kaç olduğunu bulmak için kullanırız.

Ardından:

(34) `fracCorrect = numCorrect / numel(flwrPreds)`
ile yüzde değer bulunur.

Sınıfa Göre Performans

Kayıp ve doğruluk ağın performansının genel ölçümünü verir. Ancak ağın farklı görüntü sınıflarında nasıl performans gösterdiğini araştırmak bilgilendirici olabilir.

Confusion matris (karışıklık tablosu, matrisi) işlevi, tahmin edilen sınıflandırmalar için karışıklık matrisi hesaplar. ve görüntüler.

(35) `confusionchart(knownclass, predictedclass)`

Bilinen sınıfların adının değeri

Tahmin edilen sınıfların değerleri