

Deep Learning – Image Similarity Engine

Final Report

1. Proje Tanımı ve Arka Plan:

Bu çalışmanın amacı Bil 442 - Derin Öğrenme dersi kapsamında, bir Görüntü Benzerlik Arama Motoru tasarlamaktır. Tasarlanan Görüntü Benzerlik Arama Motorunun çözmesi gereken iki problem tanımlanmıştır; (1) Bir görüntü kümesi içinde bulunan birbirine benzer görüntülerin gruplandırılması, (2) kümenin içinden, seçili herhangi bir görüntüye benzer görüntülerin bulunması. Eğitim amaçlı projede, CNN tabanlı bir autoencoder modeli oluşturularak model üzerinden latent vektörlerin (temsil vektörlerin) çıkarılması tercih edilmiştir.

Günümüzde görüntü benzerlik arama motorları; e-ticaret sitelerinde gerçek bir ürün resmine dayalı benzer ürünler bulmak, dijital doküman arşivlerinde benzer belgeleri bulmak ve doğru etiketleme yapabilmek, makine öğreniminde eğitim veri kümelerindeki kopyaları bulmak, sosyal medyada sahte hesapları ve ithal edilmiş görselleri bulma gibi alanlarda kullanılmaktadır.

Yapılan literatür taraması, “Görüntü Arama Motoru” (Image Similarity Engine), “Görüntü Arama Motoru Derin Öğrenme” (Image Similarity Engine Deep Learning) ve “İçerik Tabanlı Görüntü Benzerliği” (Content Based Image Similarity) sorgularından elde edilen 2010-2022 yılları arasında çıkarılmış çalışmaları kapsamaktadır.

Literatür taramasında, görüntü benzerlik arama motorunun tasarımı (1) görüntülerin öznitelik (temsil) vektörlerinin oluşması ve (2) Öznitelik vektörleri arasında yapılan benzerlik hesabının üstünde durulduğu gözlemlenmiştir. Vektörler arası yapılan benzerlik hesabında genelde cosine ve euclidean uzaklık metriklerinin, en benzer görüntülerin bulunması aşamasında ise en yakın komşu algoritmasının tercih edildiği gözlenmiştir. (Mishra, 2019), (Geerenstein, 2021), (Yao, 2022). Güncel çalışmalarda büyük veri setlerinde, en benzer görüntünün bulunması aşamasında ölçeklenebilirlik sorunuyla karşı karşıya kalındığı görülmüştür. Google servislerinden Vertex AI Matching Engine, IO Similarity Search, Pinecone ve Milvus kullanılarak benzer görüntüleri bulmak amacıyla milyonlarca görüntü üstünde çalışan arama algoritmasının optimize edildiği belirtilmektedir.

Görüntülerin öznitelik vektörlerinin oluşturulmasında “el ile öznitelik çıkarımı”, “Transfer Learning” ile CNN modelleri üzerinden öznitelik çıkarımı” ve “CNN tabanlı autoencoder modelleri üzerinden öznitelik çıkarımı” olmak üzere 3 farklı yöntemin kullanıldığı gözlemlenmiştir.

Bu yöntemler kronolojik sırayla incelendiğinde, 2010-2015 yılları arasında el ile öznitelik çıkarımı yönteminin daha yaygın kullanıldığı, bu yöntemde her bir görüntü için renk, doku, ve şekil bazlı özniteliklerin oluşturulduğu ve yaygın olarak “Average Mean of RGB components”, “Color Histogram”, “Wavelet Transform” yöntemlerinin kullanıldığı anlaşılmıştır (Bhoir, 2020). 2014 ve sonrasında ise transfer learning kullanarak CNN tabanlı ResNet, VGG16, VGG19 ...vb karmaşık modeller üzerinden (Sodani, 2021), (Choi, 2019), (Yao, 2022) ve CNN tabanlı autoencoder modeller üzerinden öznitelik vektörlerinin elde edilmesi yaygın hale gelmiştir. (Hou, 2017), (Pawar, 2020), (Hou, 2019).

Görüntüyü sıkıştırma veya yeni görüntü üretme işlemlerinde variational autoencoder modelleri tercih edilmiş olsa bile, görüntülerin benzerliklerinin kıyaslandığı çalışmalarda Vanilla (standart) autoencoder modelleri tercih edilmiştir. Variational modellerde loss fonksiyonu üzerinden latent vektörler (temsil vektörü) normal dağılıma zorlanmaktadır, oluşan vektörlerin latent uzayındaki dağılımları birbirine yakın hale gelmekte ve vektörler arası benzerlik hesabının sonucu değişiklik göstermektedir. (Anwar, 2021).

Bu alandaki güncel gelişmeler incelendiğinde ise, CNN tabanlı Siamese modellerinin eğitim aşamasının görüntüler arası benzerlik hesabı üzerine kurulu olduğu ve yüz tanıma alanında tercih edildiği gözlenmiştir. Siamese CNN modeli ve “triplet loss” fonksiyonu kullanılarak; iki görüntü için ayrı ayrı temsil vektörü oluşturulmaktadır ve bu iki özellik vektörü arasındaki benzerlik mesafesi kullanılarak, iki görüntünün benzer mi yoksa farklı mı olduğuna karar verilmektedir. Benzerlik/farklılık kararı “triplet loss” fonksiyonunu minimize eden treshhold değeri üzerinden verilmektedir. (Ghanmi, 2021).

Bu çalışma, eğitim amaçlı bir çalışma olduğu için, “transfer learning” kullanarak ResNet, VGG19 üzerinden görsellerin temsil vektörlerinin çıkarılması yerine, CNN tabanlı bir autoencoder modeli oluşturarak model üzerinden latent vektörlerin (temsil vektörlerin) çıkarılması tercih edilmiştir. Seçili görsele benzer görseller bulunurken en yakın komşu algoritması kullanılmıştır. Algoritmanın kullandığı benzerlik ölçütü kullanıcıdan alınmıştır (euclidean/cosine). Yapılan çalışmanın görselleştirilmesi için Görüntü Benzerlik Arama Motoru Platformu oluşturulmuştur.

2. Teknik Detaylar:

Bu başlık altında tasarım, implementasyon detaylarına ve çalışmada kullanılan teknolojilere yer verilmiştir.

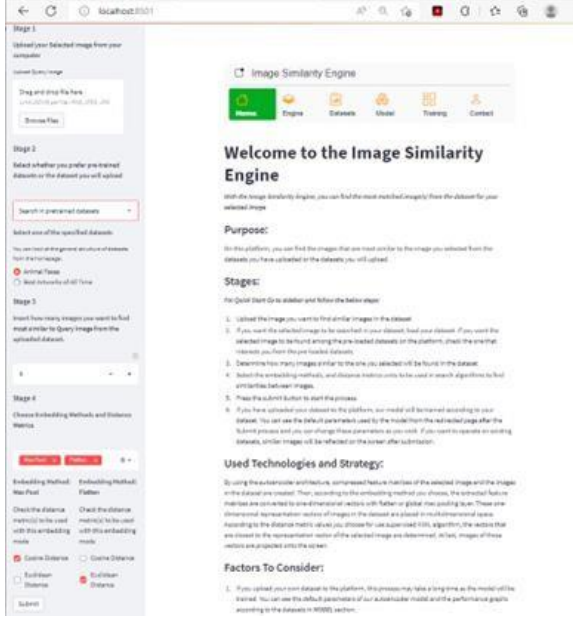
Tasarım Detayları:

Tasarımın detayları aktarılırken genelden özele bir yaklaşım izlenmiştir. İlk olarak görsel benzerlik modelinden ve bu modelin görselleştirildiği kullanıcı arayüz platformundan bahsedilmiştir. Bunun yanında, benzerlik modelinin akış şemasına ve model kapsamında oluşturulan autoencoder modellerinin tasarımına yer verilmiştir.

Modül Tanıtımı:

Çalışma *Görüntü Benzerlik Modeli* ve *Kullanıcı Arayüz Platformu* olmak üzere iki modülden oluşmaktadır.

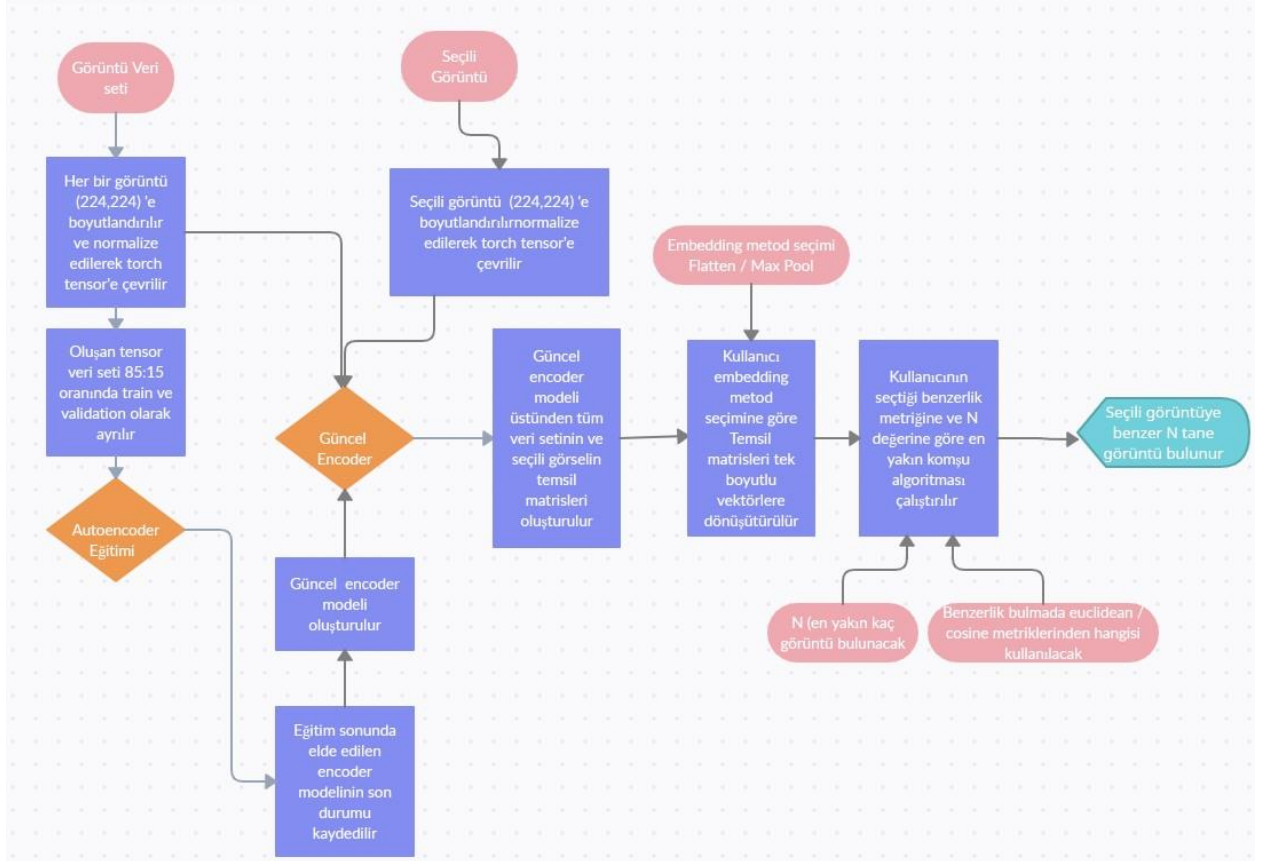
- (1) Görüntü Benzerlik Modelinde, kullanıcı tarafından belirlenen veri kümesindeki görsellerin ve kullanıcının seçtiği görselin uygun sıkıştırılmış temsil vektörleri oluşturulmaktadır. Kullanıcının belirlediği benzerlik metriği kapsamında, çıkarılan temsil vektörleri arasında en yakın komşu algoritması kullanılarak, seçilen görselin temsil vektörüne en yakın vektörler bulunmaktadır. Bulunan vektörlerin ait olduğu görseller kaydedilmektedir.



(2) Kullanıcı Arayüz Modülü ise kullanıcının Görüntü benzerlik modeli için belirlediği parametreleri girebileceği bir platformdur. Kullanıcının girdiği parametreler doğrultusunda Görüntü Benzerlik Modeli çalıştırılıp, modelin sonucunda elde edilen görseller bu platform üzerinden kullanıcıya sunulmaktadır.

Resim 1: Kullanıcı Arayüz Platformu

Görüntü Benzerlik Modeli Akış Şeması: scre



Görüntü Benzerlik Modeli İçin Tasarlanmış Autoencoder Model Yapıları

Akış şemasından da görüldüğü üzere girdi görüntüler RGB'ye çevrilerek (224,224)'e boyutlandırılmaktadır. Normalizasyon ve pytorch tensor'üne çevirme işlemi sonucunda modele girdi olarak (3,224,224) boyutlu tensörler verilmektedir.

Tablo 1: Autoencoder Modellerinin Detaylı Yapı Bilgisi

Model No	Model Açıklaması	Encoder Yapısı ve Parametre Bilgisi	Decoder Yapısı ve Parametre
Model 1	Latent (Temsil) Vektör Boyutu: (512,7,7)	<pre> ConvEncoder((conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu1): ReLU(inplace=True) (maxpool1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu2): ReLU(inplace=True) (maxpool2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu3): ReLU(inplace=True) (maxpool3): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu4): ReLU(inplace=True) (conv5): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu5): ReLU(inplace=True) (maxpool5): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) </pre> <p>Total params: 3,910,784 Trainable params: 3,910,784 Non-trainable params: 0</p>	<pre> ConvDecoder((deconv1): ConvTranspose2d(512, 512, kernel_size=(2, 2), stride=(2, 2)) (relu1): ReLU(inplace=True) (deconv2): ConvTranspose2d(512, 256, kernel_size=(2, 2), stride=(2, 2)) (relu2): ReLU(inplace=True) (deconv3): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2)) (relu3): ReLU(inplace=True) (deconv4): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2)) (relu4): ReLU(inplace=True) (deconv5): ConvTranspose2d(64, 3, kernel_size=(2, 2), stride=(2, 2)) (relu5): ReLU(inplace=True) </pre> <p>Total params: 1,738,435 Trainable params: 1,738,435 Non-trainable params: 0</p>
Model 2	Latent (Temsil) Vektör Boyutu: (1024,3,3)	<pre> ConvEncoder((conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu1): ReLU(inplace=True) (maxpool1): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu2): ReLU(inplace=True) (maxpool2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu3): ReLU(inplace=True) (maxpool3): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu4): ReLU(inplace=True) (maxpool4): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv5): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu5): ReLU(inplace=True) (conv6): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu6): ReLU(inplace=True) (maxpool6): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) </pre> <p>Total params: 8,630,400 Trainable params: 8,630,400 Non-trainable params: 0</p>	<pre> ConvDecoder((deconv1): ConvTranspose2d(1024, 512, kernel_size=(3, 3), stride=(2, 2)) (relu1): ReLU(inplace=True) (deconv2): ConvTranspose2d(512, 512, kernel_size=(2, 2), stride=(2, 2)) (relu2): ReLU(inplace=True) (deconv3): ConvTranspose2d(512, 256, kernel_size=(2, 2), stride=(2, 2)) (relu3): ReLU(inplace=True) (deconv4): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2)) (relu4): ReLU(inplace=True) (deconv5): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2)) (relu5): ReLU(inplace=True) (deconv6): ConvTranspose2d(64, 3, kernel_size=(2, 2), stride=(2, 2)) (relu6): ReLU(inplace=True) </pre> <p>Total params: 6,457,539 Trainable params: 6,457,539 Non-trainable params: 0</p>
Model 3	Latent (Temsil) Vektör Boyutu: (1024,3,3) Herhangi bir Max Pool katmanı kullanılmamıştır	<pre> ConvEncoder((conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu1): ReLU(inplace=True) (conv2): Conv2d(64, 64, kernel_size=(2, 2), stride=(2, 2)) (relu2): ReLU(inplace=True) (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu3): ReLU(inplace=True) (conv4): Conv2d(128, 128, kernel_size=(2, 2), stride=(2, 2)) (relu4): ReLU(inplace=True) (conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu5): ReLU(inplace=True) (conv6): Conv2d(256, 256, kernel_size=(2, 2), stride=(2, 2)) (relu6): ReLU(inplace=True) (conv7): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu7): ReLU(inplace=True) (conv8): Conv2d(512, 512, kernel_size=(2, 2), stride=(2, 2)) (relu8): ReLU(inplace=True) (conv9): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu9): ReLU(inplace=True) (conv10): Conv2d(512, 512, kernel_size=(2, 2), stride=(2, 2)) (relu10): ReLU(inplace=True) (conv11): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu11): ReLU(inplace=True) (conv12): Conv2d(1024, 1024, kernel_size=(2, 2), stride=(2, 2)) (relu12): ReLU(inplace=True) </pre> <p>Total params: 15,268,416 Trainable params: 15,268,416 Non-trainable params: 0</p>	<pre> ConvDecoder((deconv1): ConvTranspose2d(1024, 512, kernel_size=(3, 3), stride=(2, 2)) (relu1): ReLU(inplace=True) (deconv2): ConvTranspose2d(512, 512, kernel_size=(2, 2), stride=(2, 2)) (relu2): ReLU(inplace=True) (deconv3): ConvTranspose2d(512, 256, kernel_size=(2, 2), stride=(2, 2)) (relu3): ReLU(inplace=True) (deconv4): ConvTranspose2d(256, 128, kernel_size=(2, 2), stride=(2, 2)) (relu4): ReLU(inplace=True) (deconv5): ConvTranspose2d(128, 64, kernel_size=(2, 2), stride=(2, 2)) (relu5): ReLU(inplace=True) (deconv6): ConvTranspose2d(64, 3, kernel_size=(2, 2), stride=(2, 2)) (relu6): ReLU(inplace=True) </pre> <p>Total params: 6,457,539 Trainable params: 6,457,539 Non-trainable params: 0</p>
Model 4	Latent (Temsil) Vektör Boyutu: (1024,2,2) Diğer modellerin aksine Convolutional ve Max Pool katmanlarında (3,3)'lük filtreler kullanılmıştır.	<pre> ConvEncoder((conv1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu1): ReLU(inplace=True) (maxpool1): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False) (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu2): ReLU(inplace=True) (maxpool2): MaxPool2d(kernel_size=(2, 2), stride=2, padding=0, dilation=1, ceil_mode=False) (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu3): ReLU(inplace=True) (maxpool3): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False) (conv4): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu4): ReLU(inplace=True) (maxpool4): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False) (conv5): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu5): ReLU(inplace=True) (conv6): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1)) (relu6): ReLU(inplace=True) (maxpool6): MaxPool2d(kernel_size=(3, 3), stride=2, padding=0, dilation=1, ceil_mode=False) </pre> <p>Total params: 8,630,400 Trainable params: 8,630,400 Non-trainable params: 0</p>	<pre> ConvDecoder((deconv1): ConvTranspose2d(1024, 512, kernel_size=(3, 3), stride=(2, 2), output_padding=(1, 1)) (relu1): ReLU(inplace=True) (deconv2): ConvTranspose2d(512, 512, kernel_size=(3, 3), stride=(2, 2)) (relu2): ReLU(inplace=True) (deconv3): ConvTranspose2d(512, 256, kernel_size=(3, 3), stride=(2, 2)) (relu3): ReLU(inplace=True) (deconv4): ConvTranspose2d(256, 128, kernel_size=(3, 3), stride=(2, 2)) (relu4): ReLU(inplace=True) (deconv5): ConvTranspose2d(128, 64, kernel_size=(3, 3), stride=(2, 2)) (relu5): ReLU(inplace=True) (deconv6): ConvTranspose2d(64, 3, kernel_size=(3, 3), stride=(2, 2), output_padding=(1, 1)) (relu6): ReLU(inplace=True) </pre> <p>Total params: 8,629,379 Trainable params: 8,629,379 Non-trainable params: 0</p>

İmplementasyon Detayları:

İmplementasyon detayları (1) Girdiyi İşleme, (2) Model İmplementasyonu, (3) Temsil Vektörü Oluşturma, (4) Benzer Resimleri Bulma başlıkları altında kronolojik ve sistem akışına paralel şekilde anlatılmıştır. Bu kısımda yer verilen model kayıpları, modellerin [dataset 1](#) üzerinde çalıştığı takdirde gösterdikleri kayıp değerleridir.

Girdiyi İşleme:

Hafıza ve GPU desteği kısıtlı olduğu için veri setindeki görüntüler ve seçili görüntü RGB görüntülere çevrilip (224,224) boyutuna resize edilmiştir. Bunun yanında tüm görüntüler için normalizasyon ve pytorch tensör gösterimine (*batch size, channel,width,height*) çevirme işlemleri yapılmıştır.

Model İmplementasyonu:

Çalışmada en iyi performansı gösteren (en kaliteli temsil matrislerini çıkaran) modeli bulabilmek amacıyla 6 farklı model oluşturulmuştur. Bu raporda oluşturulan modellerden en başarılı 4 tanesinin detaylı implementasyonuna yer verilmiştir. Modellerin detaylı yapıları "[Görüntü Benzerlik Modeli İçin Tasarlanmış Autoencoder Model Yapıları](#)" başlığı altından incelenebilmektedir.

- İlk olarak (Oke, 2020), (Chaitanyanarava, 2020) ve (Lippe, 2022) çalışmaları referans alınarak 2 farklı model oluşturulmuştur. Bu modeller içerisinde en iyi performansı sergileyen model seçilmiştir.
- Seçilen modelin, aktivasyon fonksiyonları ReLu'ya çevrilerek, learning rate parametresi, 0.001 olarak güncellenerek, epoch sayısı 40 olarak değiştirilerek [Model 1](#) oluşturulmuştur.
- Model 1' in eğitim aşamasında 0.008 - 0.006 değer aralığında kayıp alınmıştır ancak çıkarılan temsil vektörlerinin kaliteli olmadığı, bundan dolayı sistemin benzer resimleri bulma oranının düşük olduğu gözlemlenmiştir. Detaylı sonuçlar "[İyi Performans Gösteren Modellerin Benzerlik Başarımları](#)" başlığı altından ulaşılabilir
- Bu durumu düzeltmek amacıyla Model 1'in encoder modülü biraz daha geliştirilmiştir. Encoder Modülüne 3x3'lük convolution katmanı ve 2x2'lik stride =2 olan bir Max Pool katmanı eklenmiştir. Decoder modülüne ise bu işleme karşılık gelen ConvTranspose katmanı eklenmiştir. Model 1 sonunda elde edilen (512,7,7) büyüklüğündeki temsil matrisi (1024,3,3) temsil matrisine dönüştürülmüş ve [Model 2](#) oluşturulmuştur.
- Oluşturulan Model 2 için eğitim aşamasında 0.0018 – 0.001 değer aralığında kayıp alınmıştır. Ancak Model 2 sonucunda elde edilen temsil vektörlerinin kalitesi Model 1'in sonucunda elde edilen temsil vektörlerine kalitesine kıyasla daha iyidir. Bu durumdan dolayı Model 2 ile oluşturulan sistemle Model 1'le oluşturulan sisteme kıyasla daha benzer resimleri bulunabilinmiştir. Performans kıyaslaması "[İyi Performans Gösteren Modellerin Eğitim Parametreleri ve Eğitim Sonuçları](#)" başlığı altından bulunabilir
 - Model 2'nin eğitim aşamasında elde edilen kaybın büyük olmasındaki nedeninin Mean Square Error kayıp fonksiyonunun, Max Pool katmanı ve çift boyutlu kernel seçiminin neden olduğu piksel shift için hassas olması olduğu öğrenilmiştir (Lippe, 2022). Bunun üzerinde Max Pool katmanlarının ve, çift boyutlu Kernel seçiminin değiştirildiği 2 farklı model oluşturulmuştur. [Model 3](#)'de (Volodymyr Turchenko, 2017) çalışmasında bahsedildiği üzere tüm max_pool katmanları convolutional katmanlara çevrilmiştir. Model 4'de ise tüm max pool ve convolutional katmanlarında kullanılan filtreler 3x3'lük filtrelere dönüştürülmüştür.

- Bu dört model içerisinde en kaliteli vektör matrislerinden biri olan Model 2'nin kaybını azaltmak için, learning rate parametresi üzerinden eğitim aşamasındaki her batch için *Cosine annealing*, *Cosine Annealing Warm Restarts* ve *ReduceLROnPlateau* learning rate düzenleyicileri uygulanmıştır. Bunun yanında MSE (Mean Squared Error) kayıp fonksiyonu yerine Structural Similarity Index'in kullanıldığı (SSIM) kayıp fonksiyonu oluşturulmuş ve model bu kayıp fonksiyonu ile eğitilmiştir.

Temsil Vektörü Oluşturma:

Çalışmada autoencoder modeller sonucunda elde edilen temsil matrislerinden temsil vektörleri oluşturmak için (1) Flatten, (2) Max pool olmak üzere iki farklı metot kullanılmıştır. Flatten metodunda güncel encoderdan elde edilen temsil matrisleri Flatten katmanı üzerinden herhangi bir piksel kaybına uğramadan tek boyutlu vektöre dönüştürülmüştür. Max Pool metodunda ise elde edilen temsil matrisleri üzerinde *global max pooling* uygulanmış, her channel'ın en büyük (en ayırt edici) değeri seçilerek channel sayısı boyutunda temsil vektörü oluşturulmuştur.

En Benzer Resimleri Bulma,

Veri setindeki görsellerin temsil vektörleri arasından, seçili resmin temsil vektörüne en çok benzeyen vektörlerin tespiti için, Local Sensitive Hashing, En yakın komşu ve K-means algoritmaları kullanılmıştır. Bu 3 algoritmanın sonuçlarının seçilen uzaklık metriğine göre (cosine/euclidean) paralellik gösterdiği tespit edilmiştir.

Kullanılan Teknolojiler:

Çalışmada kullanılan yazılım dili, Python 3.8'dir, Görüntü Benzerlik Modelinin tasarımında Pytorch 1.11 Framework'u, Kullanıcı Arayüz Platformu'nun tasarımında Streamlit 1.10.0 Framework'u kullanılmıştır. Görüntü Benzerlik Modelinde yapılan tüm işlemlerde N Cvidia GeForce GTX 1650 4GB ekran kartı ile Intel i7-9750H işlemcisi kullanılmıştır. C

Tablo 2: Görüntü Benzerlik Modelinde Kullanılan Kütüphaneler

Framework	Version
pillow	9..0.1
numpy	1.21.5
matplotlib	3.5.1
scikit-learn	1.1.1
tensorboard	2.9.0
scipy	1.8.1
lshash3	0.0.8
opencv-python	4.6.0.66

Tablo 3: Eğitim Aşamasında Kullanılan Veri Setleri

Veri Seti Adı	Görüntü Sayısı	Veri seti Tanımı	Path
<i>Animal Faces</i>	4738	Altı tür yüksek kaliteli hayvan yüzü içermektedir	\Proje_v0\dataset_v1
<i>Best Artworks of All Time</i>	8355	Elli farklı ressamın ait resim 200 civarında renkli veya karakalem resimlerini içermektedir.	\Proje_v0\dataset_v2

Autoencoder modelinde kullanılan kayıp fonksiyonunun performans analizinde MSE fonksiyonu yerine SSIM tabanlı kayıp fonksiyonu oluşturulmuştur. Bu fonksiyon oluşturulurken Gongfan Fang, Zhejiang tarafından oluşturulan SSIM implementasyonu hakları korunarak (*Copyright 2020 by Gongfan Fang, Zhejiang University*) kopyalanmıştır.

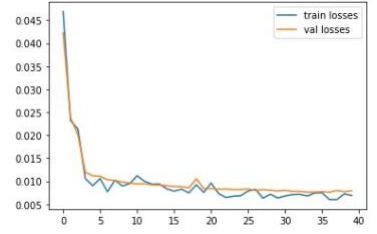
3. Deneyler ve Sonuçlar

Bu başlık altında, çalışma sırasında oluşturulan ve diğerlerine kıyasla daha iyi sonuç veren sistemlerde kullanılan 4 modelin; eğitim sırasında kullandıkları parametreler, kayıp grafikleri ve benzerlik başarımları verilmiştir.

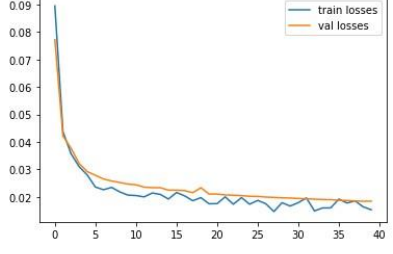
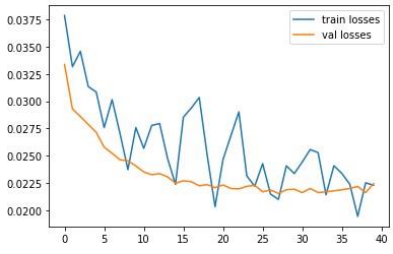
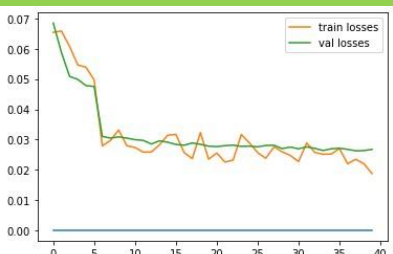
Aşağıda detaylı eğitim bilgisi verilen modellerin eğitimde kullandıkları Learning Rate parametresinin başlangıç değeri 0.001'dir. Bunun yanında, eğitimin gerçekleştiği epoch sayısı 40 ve eğitim sırasında kullanılan batch boyutu 32'dir. Bunun yanında detayları verilen eğitim sonuçları, modellerin dataset 1 üzerinde eğitilmelerinden sonra ulaşılan sonuçlar ise mavi dataset 2 üzerinde eğitilmelerinden sonra ulaşılan sonuçlar ise kırmızı renkte gösterilmiştir.

İyi Performans Gösteren Modellerin Eğitim Parametreleri ve Eğitim Sonuçları

Tablo 4: İyi performans gösteren modellerin eğitim parametreleri ve kayıp fonksiyonları

Model	Learning Scheduler	Loss Fonksiyonu	Min Loss Değeri: (Veri Seti 1 ve Veri Seti 2 için)	Loss Grafiği
Model 1- latent vektör (512,7,7)	X	MSE	Iteration 1: 0.00662 Iteration 2: 0.007578	

Model	Learning Scheduler	Loss Fonksiyonu	Min Loss Değeri: (Veri Seti 1 ve Veri Seti 2 için)	Loss Grafiği
Model 2- latent vektör (1024,3,3),	X	MSE	Iteration 1: 0.01402 Iteration 2: 0.01688 Iteration 3: 0.01025 Iteration 1: 0.00863	
	Cosine Annealing Warm Restarts T_0=5, T_mult=1, eta_min=0.0001,	MSE	Iteration1: 0.01415 Iteration2: 0.01705	
	Cosine Annealing Warm Restarts T_0=10, T_mult=1, eta_min=0.0001,	MSE	Iteration 1: 0.0107 Iteration 2: 0.01265	
	ReduceLROnPlateau mode='min', factor=0.2, patience=2,	MSE	Iteration 1: 0.01028 Iteration 2: 0.01308	
Model 3 – LV: (1024,3,3), sadece convolutional katmanlar kullanılmıştır	X	MSE	0.01087	

	ReduceLROnPlateau mode='min', factor=0.2, patience=2, threshold=1e-4, min_lr=0.00005	MSE	0.01823	
Model	Learning Scheduler	Loss Fonksiyonu	Min Loss Değeri: (Veri Seti 1 ve Veri Seti 2 için)	Loss Grafiği
Model 4- LV: (1024,2,2) 3x3 boyutlu convolutional ve max pool katmanları kullanılmıştır	X	MSE	0.0216	
	ReduceLROnPlateau mode='min', factor=0.2, patience=2, threshold=1e-4, min_lr=0.00005	MSE	0.026 0,026	

İyi Performans Gösteren Modellerin Benzerlik Başarımları :

Benzerlik başarımları ölçülürken dataset 2 yerine dataset 1 kullanılmıştır. Bunun nedeni dataset 1 içinde daha az çeşitlikte görselin bulunmasıdır. Bunun yanında, dataset 1'in içerdiği veriler arasındaki benzerliğin dataset 2'nin içerdiği verilere kıyasla, insan gözüyle daha rahat ayırt edilebilir olmasıdır.

Sistemlerin başarımları, sistem sonucunda elde edilen 8 görüntü içinden kaç tanesinin seçili görüntüye benzer görüntü olduğuna göre hesaplanmıştır. Eğer sistem sonucunda elde edilen hayvan görüntülerinin çoğu seçili görüntüyle benzeyen "aynı cins" hayvan görüntüleri içeriyorsa sistem başarılıdır. Kabul edilmiştir.

Tablolardaki görsel sonuçlara verilen pathler kullanılarak ulaşılabilir

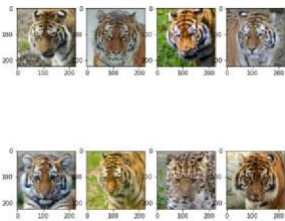
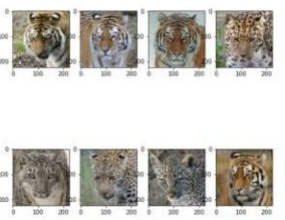
Seçili Görüntü



Tablo 5: Modellerin Seçili Görüntüye Göre Benzerlik Başarımları

Model	Learning scheduler	Embedding Method	Distance Metric	Results	Plots
Model 1- latent vektör (512,7,7)	X	Flatten	Cosine	6/8	
		Flatten	Euclidean	3/8	
		Max Pool	Cosine	6/8	
		Max Pool	Euclidean	7/8	

Model	Learning scheduler	Embedding Method	Distance Metric	Results	Plots
-------	--------------------	------------------	-----------------	---------	-------

Model 2- latent vektör (1024,3,3),	X	Flatten	Cosine	7/8	
		Flatten	Euclidean	5/8	
		Max Pool	Cosine	7/8	
		Max Pool	Euclidean	7/8	
	Cosine Annealing Warm Restarts T_0=5, T_mult=1, eta_min=0.0001,	Flatten	Cosine	5/8	Path: \Proje_v0\Result_dataset1 \Model4\CAW\caw_T_max 5
		Flatten	Euclidean	2/8	
		Max Pool	Cosine	1/8	
		Max Pool	Euclidean	1/8	
		Flatten	Cosine	5/8	Path: \Proje_v0\Result_dataset1 \Model4\CAW\caw_T_max 10
		Flatten	Euclidean	2/8	
		Max Pool	Cosine	5/8	
		Max Pool	Euclidean	5/8	
		Flatten	Cosine	5/8	Path:

Model	Learning scheduler	Embedding Method	Distance Metric	Results	Plots
	ReduceLROnPlate au mode='min', factor=0.2, patience=2,	Flatten	Euclidean	4/8	\Proje_v0\Result_dataset1 \Model4\ReduceLR
		Max Pool	Cosine	6/8	
		Max Pool	Euclidean	5/8	
Model 3 – LV: (1024,3,3),sadece convolutional katmanlar kullanılmıştır	X	Flatten	Cosine	2/8	Path: \Proje_v0\Result_dataset1 \Model5
		Flatten	Euclidean	1/8	
		Max Pool	Cosine	4/8	
		Max Pool	Euclidean	2/8	
	ReduceLROnPlate au mode='min', factor=0.2, patience=2, threshold=1e-4, min_lr=0.00005	Flatten	Cosine	1/8	Path: \Proje_v0\Result_dataset1 \Model5\ReduceLR
		Flatten	Euclidean	1/8	
		Max Pool	Cosine	1/8	
		Max Pool	Euclidean	1/8	
Model 4- LV: (1024,2,2) 3x3 boyutlu convolutional ve max pool katmanları kullanılmıştır	X	Flatten	Cosine	7/8	Path: \Proje_v0\Result_dataset1 \Model6
		Flatten	Euclidean	7/8	
		Max Pool	Cosine	2/8	
		Max Pool	Euclidean	4/8	
	ReduceLROnPlate au mode='min', factor=0.2, patience=2, threshold=1e-4, min_lr=0.00005	Flatten	Cosine	7/8	Path: \Proje_v0\Result_dataset1 \Model6\ReduceLR
		Flatten	Euclidean	7/8	
		Max Pool	Cosine	6/8	
		Max Pool	Euclidean	8/8	

En başarılı sistemler seçilirken, sistemlerin 4 farklı Embedding Method ve Distance Metric ikllisinde iyi performans göstermesine bakılmıştır. İyi performans gösteren sistemler Tablo 4 ve Tablo 5 de yeşil renkli olarak belirtilmiştir.

Model 4 ve Model 6 Similarity Index'in kullanıldığı (SSIM) kayıp fonksiyonu ile eğitildiğinde test kümesinde minimum %20'lik kayıp gözlenmiştir. Bundan dolayı bu fonksiyonun kullanıldığı sistemin detayları yukarıdaki tablolarda verilmemiştir.

4. Sonuç

Yapılan deneyler analiz edildiğinde bu çalışma kapsamında oluşturulan Model 4'ün ReduceLROnPlateue düzenleyicisi kullanılan versiyonu ile Model 2'nin herhangi bir düzenleyici kullanılmadığındaki versiyonunun, diğer modellere kıyasla yüksek başarımlar gösterdiği gözlenmiştir. Bu durumda, latent (temsil) vektör boyutu küçüldükçe ve channel sayısı arttıkça, uygun eğitim parametreleriyle beraber daha kaliteli modellerin ortaya çıktığı görülmüştür. Bunun yanında autoencoder modeller kullanılarak yapılan benzerlik çalışmalarında, modellerden elde edilen kayıp fonksiyonlarının modellerin benzerlik başarımları hakkında bilgi vermediği anlaşılmıştır. Son olarak, yapılan deneyler incelendiğinde, modellerin genelinde Max Pool metodu kullanılarak ve Cosine uzaklık metriği seçilerek diğer <Embedding Method, Distance Metric> kombinasyonlara kıyasla daha yüksek benzerlik başarımları elde edilebileceği gözlenmiştir.

Referanslar

Anwar Aqeel [Çevrimiçi] // towards data science. - 2021. -

[https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-](https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2#:~:text=This%20is%20where%20the%20Autoencoder,dimensional%20space%2C%20essentially%20achieving%20compression..)

[ed7be1c038f2#:~:text=This%20is%20where%20the%20Autoencoder,dimensional%20space%2C%20essentially%20achieving%20compression..](https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2#:~:text=This%20is%20where%20the%20Autoencoder,dimensional%20space%2C%20essentially%20achieving%20compression..)

Bhoir Smita V. A Review on Recent Advances in Content-Based Image Retrieval [Dergi]. - 2020. - Cilt Library Philosophy and Practice (e-journal). 5617..

Chaitanyanarava Image similarity model [Çevrimiçi] // Medium. - 2020. -

<https://medium.com/analytics-vidhya/image-similarity-model-6b89a22e2f1a>.

Choi Hyewon VISE: vehicle image search engine with traffic camera [Dergi]. - 2019. - Cilt

<https://doi.org/10.14778/3352063.3352080>.

Geerenstein van Image Search Engine for Digital History: A deep learning approach [Dergi]. - 2021.

- Cilt <http://resolver.tudelft.nl/uuid:f1a2902b-14be-416c-ae1a-ce4f179a0425>.

Ghanmi CheckSim: A Reference-Based Identity Document Verification by Image Similarity Measure.

[Dergi]. - 2021. - Cilt Lecture Notes in Computer Science, vol 12916. Springer, Cham.

https://doi.org/10.1007/978-3-030-86198-8_30.

Hou Xianxu Deep Feature Consistent Variational Autoencoder [Dergi]. - 2017. - Cilt IEEE Winter

Conference on Applications of Computer Vision (WACV), 2017, pp. 1133-1141, doi:

[10.1109/WACV.2017.131..](https://doi.org/10.1109/WACV.2017.131..)

Hou Xianxu Improving variational autoencoder with deep feature consistent and generative adversarial training [Dergi]. - 2019. - Cilt <https://doi.org/10.1016/j.neucom.2019.03.013>.

Lippe Phillip Deep Autoencoders [Çevrimiçi]. - 2022. - https://colab.research.google.com/github/PytorchLightning/lightningtutorials/blob/publication/.notebooks/course_UvA-DL/08-deepautoencoders.ipynb#scrollTo=45614b53.

Mishra Richa Deep learning based search engine for biomedical images using convolutional neural networks [Dergi]. - 2019. - Cilt Multimed Tools Appl 80, 15057–15065 (2021). <https://doi.org/10.1007/s11042-020-10391-w>.

Oke Aditya Image Similarity Search in PyTorch [Çevrimiçi] // Medium. - 2020. - <https://medium.com/pytorch/image-similarity-search-in-pytorch-1a744cf3469>.

Pawar Aashay Evaluation of autoencoder for CBIR system in deep learning [Dergi]. - 2020. - Cilt IEEE 17th India Council International Conference (INDICON), 2020, pp. 1-4, doi: 10.1109/INDICON49873.2020.9342239..

Sodani Abhigya Scalable Reverse Image Search Engine for NASAWorldview [Dergi]. - 2021. - Cilt arXiv:2108.04479 [cs.CV].

Volodymyr Turchenko Eric Chalmers, Artur Luczak A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe [Dergi]. - 2017. - Cilt arXiv:1701.04949.

Yao Image Search Engine by Deep Neural Networks [Dergi]. - 2022. - Cilt In J. Louveaux, & F. Quitin (Eds.), 42nd WIC Symposium on Information).