EGE ÜNİVERSİTESİ

EÜ MÜHENDİSLİK FAKÜLTESİ

LİSANS TEZİ

BİRLEŞİK TRANSFORMER AĞI İLE ÇOK AMAÇLI MODEL

Mehmet ÜNLÜ

Tez Danışmanı : Prof. Dr. Aybars UĞUR

Bilgisayar Mühendisliği Anabilim Dalı

Sunuş Tarihi : 20/06/2023

Bornova-İZMİR
2023

ÖZET
BİRLEŞİK TRANSFORMER AĞI İLE ÇOK AMAÇLI MODEL

ÜNLÜ, Mehmet

Lisans Tezi, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Prof. Dr. Aybars UĞUR
Haziran 2023, 46 sayfa

Transformer, özellikle dil anlama için popüler olarak kullanılan bir sinir ağı mimarisidir. Görüntü, metin, video vb. gibi çeşitli modaliteleri içeren görevler için kullanılabilecek genişletilmiş ve birleştirilmiş bir mimari sunuyoruz. Zamansal girdi dizisine karşılık gelen gizli durumlara ek olarak girdinin uzamsal boyutunu öğrenmeyi sağlayan bir uzamsal-zamansal önbellek mekanizması öneriyoruz. Önerilen mimari ayrıca tek bir modelin birden fazla girdi modalitesine sahip görevleri ve eş zamanlı olmayan çoklu görev öğrenmeyi desteklemesini sağlar. Bu modelin tek bir örneği aynı anda resim altyazısı ekleme, görsel soru yanıtlama ve video etkinliği tanıma görevlerini yerine getirmeyi öğrenebilir. Bu üç görevi birlikte eğitmenin, ayrı ayrı eğitmeye kıyasla performansı korurken yaklaşık üç kat sıkıştırılmış modelle sonuçlandığını gösteriyoruz. Ayrıca, bazı modaliteler üzerinde önceden eğitilmiş bu sinir ağını kullanmanın, video altyazısı ve video soru cevaplama gibi görünmeyen görevleri öğrenmeye yardımcı olduğunu gösteriyoruz. Bu, modelde bulunan uzamsal-zamansal önbellek üzerindeki öz-dikkat mekanizmasının genelleme kapasitesini göstermektedir. Modaliteler arasındaki temsiller güçlendikçe genelleme kapasitesi de artacaktır.

Anahtar kelimeler: Çok amaçlı modeller, Yapay Sinir Ağları, Transformer, Çok Modlu Makine Öğrenmesi, Çok Görevli Öğrenme, Uzamsal-zamansal Mekanizma, Dikkat Mekanizması, Derin Öğrenme, Pytorch.

ABSTRACT
MULTIPURPOSE MODEL WITH UNIFIED TRANSFORMER

ÜNLÜ, Mehmet

Bachelor in Computer Eng.
Supervisor: Prof. Dr. Aybars UĞUR
June 2023, 46 pages

We introduce an expanded and integrated neural network structure, a type of architecture known as Transformer, which is renowned for language comprehension. This structure can be implemented for a wide range of tasks that incorporate different mediums such as images, text, and video. Our design integrates a spatio-temporal cache mechanism that enhances the learning of both the spatial aspects of the input and the hidden states tied to the temporal input sequence. The innovative architecture permits one model to concurrently support tasks with various input modalities and manage asynchronous multitask learning. A single instance of our model can concurrently be trained to execute tasks like image captioning, visual question answering, and video event identification. Our research shows that training these three tasks collectively results in a model that's nearly three times more compact, with no compromise on performance when compared to training them separately. Furthermore, we illustrate that employing this neural network, pre-trained on specific modalities, aids in learning unseen tasks like video captioning and video question answering. This proves the generalizing capabilities of the self-attention mechanism over the spatio-temporal cache incorporated in the model. As the representations across modalities enhance, so does the capacity for generalization.

Keywords: Multipurpose Models, Transformer, Multimodal Machine Learning, Multitask Learning, Artificial Neural Networks, Spatio-temporal Mechanism, Deep Learning, Attention Mechanism, Pytorch.

TEŞEKKÜR

Lisans tezimizin hazırlanmasında, tezimiz ile ilgilenen, bize her aşamada yol gösteren; desteğini ve yardımlarını yoğun çalışma programına rağmen esirgemeyen danışmanımız Sayın Prof. Dr. Aybars UĞUR'a teşekkürü borç bilirim.

İÇİNDEKİLER

Sayfa

ŞEKİLLER DİZİNİ

Şekil Sayfa

TABLOLAR DİZİNİ

Tablo Sayfa

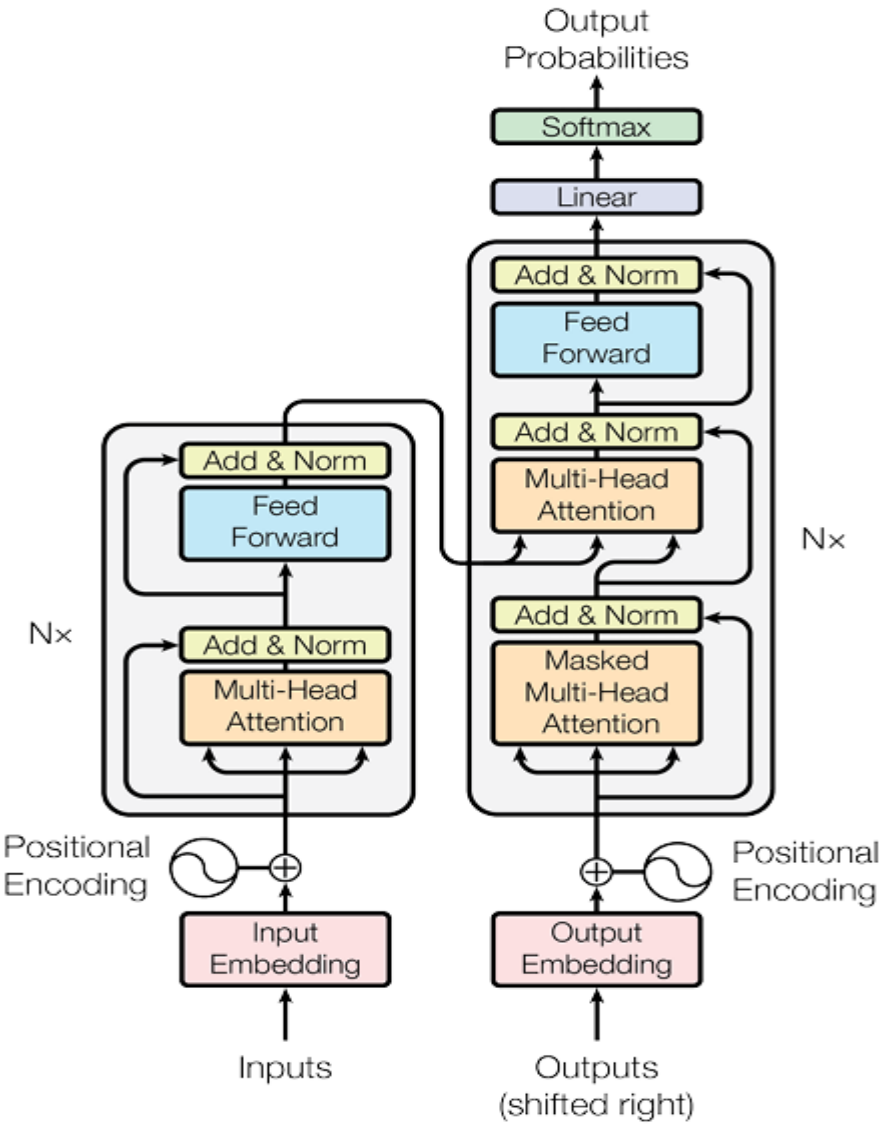
KISALTMALAR DİZİNİ

Kısaltmalar

UniT	Unified Transformer (birleşik transformer)
LSTM	Uzun Kısa Süreli Bellek
NLP	Natural Language Processing (Doğal dil işleme)
CV	Computer Vision (Bilgisayarlı görü)
GPU	Graphic Processing Unit (Grafik işlem birimi)
SOTA	State-of-the-art (son teknoloji, son gelişmeler)
CNP	Merkezi Sinir İşlemcisi
MTL	Çok Görevli Öğrenme

1. GİRİŞ

Transformer [1], doğal dil ile ilgili görevler başta olmak üzere herhangi bir dizi dönüştürme görevi için en iyi performansı gösteren modellerden biridir. Özgün olarak tek bir görevi gerçekleştirmek üzere tasarlanmıştır. Aslında, geliştirilmiş olan genel derin öğrenme yapıları [19, 3, 4], tek bir görevi ve belirli bir girdi alanını (örneğin resim, metin veya ses) oldukça iyi bir şekilde öğrenebilir. Bu modellerle genellikle, eğitilmiş ağın genelleme yeteneğine güveniriz, bu sayede görülmemiş örnekler üzerinde de performans garantilenir. Transfer öğrenme, modelin benzer girdi alanına sahip ilgili bir görevi öğrenmesi için sıkça kullanılan bir paradigmadır.



Şekil 1.1 Transformer model mimarisi [1]

Verinin etkili temsillerini öğrenebilme yeteneklerinden dolayı, sinir ağlarının bu zorluklar karşısındaki başarısı bilinmektedir. Örneğin şekil 1.1'de gösterildiği gibi, Transformer'daki öz-yinelemeli mekanizma, sıralı verilerdeki global zamansal bağımlılığı çok iyi yakalayabilir. Doğal olarak, Transformer gibi yapıları, birden çok girdi alanından ortak temsiller öğrenmeye ve bu temsillere dayanarak bir dizi görevi eşzamanlı olarak gerçekleştirmeye genişletebileceğimiz sorusu ortaya çıkar.

Çeşitli girdi alanları arasında değişken görevleri çözmeyi öğrenen çoklu görev modelleri üzerine yapılan araştırmalar yeni değildir. [5] numaralı çalışma, ses ve video modaliteleri arasında ortak bir temsil öğrenebilen bir yapıyı göstermektedir. Benzer şekilde, [6] numaralı çalışmada bir dizi dil işleme görevini desteklemek için bir evrişimli yapı tasarlanmıştır. Tek bir modaliteyi modellemenin yanı sıra, transformer modelleri ayrıca görsel soru cevaplama gibi ortak görüş ve dil akıl yürütme görevlerinde de güçlü bir performans sergiler (örneğin, [8, 9, 10, 11). Ancak, bu yapıların çoğu, belirli girdi alanları ile bilinen bir dizi görevi öğrenmek üzere tasarlanmıştır.

Ancak, transformerların belirli alanlara uygulanması konusundaki yukarıdaki başarılara rağmen, transformerlarla farklı görevler arasında çok fazla bağlantı kurma çabası olmamıştır. Transformerların başarısına tanıklık ettikten sonra, doğal olarak çeşitli sorular ortaya çıkar: bir transformer modeli, metinsel giriş üzerinde doğal dil çıkarımı için eğitildiğinde resimler üzerinde nesne tespiti de yapabilir mi, ya da bir transformerlara dayalı bir resim sınıflandırıcı metinsel çıkarımları kontrol edebilir mi? Genel olarak, çeşitli alanlardaki görevleri eşzamanlı olarak ele alabilen tek bir model oluşturmak mümkün müdür? Önceki çalışmalar bu tür sorunları çözmeye çalışmıştır, ancak sınırlı bir kapsamda:

• sadece tek bir alanın veya belirli çok modlu alanların görevlerine uygulanmıştır; ViT [12] ve DETR [13] yalnızca görüşe dayalı görevlere

odaklanırken, BERT [14] ve türevi çalışmalar sadece dil görevlerini ele alırken, VisualBERT, VILBERT [8, 15] ve diğer çok modlu transformerlar sadece görüş ve dilin belirli çok modlu alanında çalışır.

• her bir görev için görevlere özel ince ayarları içerir, görevler arasında paylaşılan hiçbir parametreyi kullanmaz, genellikle N görev için N katı parametreye sahip olur, örneğin, BERT ile her bir görev için ayrı ayrı bir modeli ince ayar yapmak zorundadır.

• genellikle sert kodlanmış eğitim stratejileriyle tek bir alanın ilgili veya benzer görevleri üzerinde çoklu görev yapar; örneğin, T5 [8] yalnızca dil alanındaki görevler üzerinde çalışırken, VILBERT-MT [9] yalnızca ilgili görüş ve dil görevleri üzerinde çalışır.

Bu çalışmada, resim ve/veya metinleri giriş olarak alan ve görsel algıdan doğal dil anlamaya ve ortak görüş ve dil akıl yürütmeye kadar bir dizi görev üzerinde ortaklaşa eğitilen bir Unified Transformer (UniT) [2] modeli oluşturuyoruz. Bu yapı, tek bir modelin birden çok girdi modalitesi ile görevleri desteklemesini ve asenkron çoklu görev öğrenmesini mümkün kılar. Gerçek yaşam verilerinin, örneğin resim, metin, konuşma, video vb. genellikle mekansal ve zamansal bileşenlerin birleşimi olduğunu düşünüyoruz. Bu yüzden, girdi verilerinin mekansal (uzay) ve zamansal (zaman) boyutları üzerinden ortak bir temsil öğrenmek için bir mekansal-zamansal önbellek mekanizması kullanıyoruz. Genelleştirilmiş bir encode() fonksiyonunu kullanarak, her bir girdi alanı için mekansal-zamansal temsili işleyebilir ve saklayabilir ve ardından çeşitli görevler üzerinden tahminleri decode() yapabilir. Deneylerimizde kelime türü etiketleme, resim altyazılama, görsel soru cevaplama ve video etkinlik tanıma gibi birden çok alanı kapsayan bir dizi görevi çözmek için eğitiyoruz.

2. ÖNCEKİ ÇALIŞMALAR

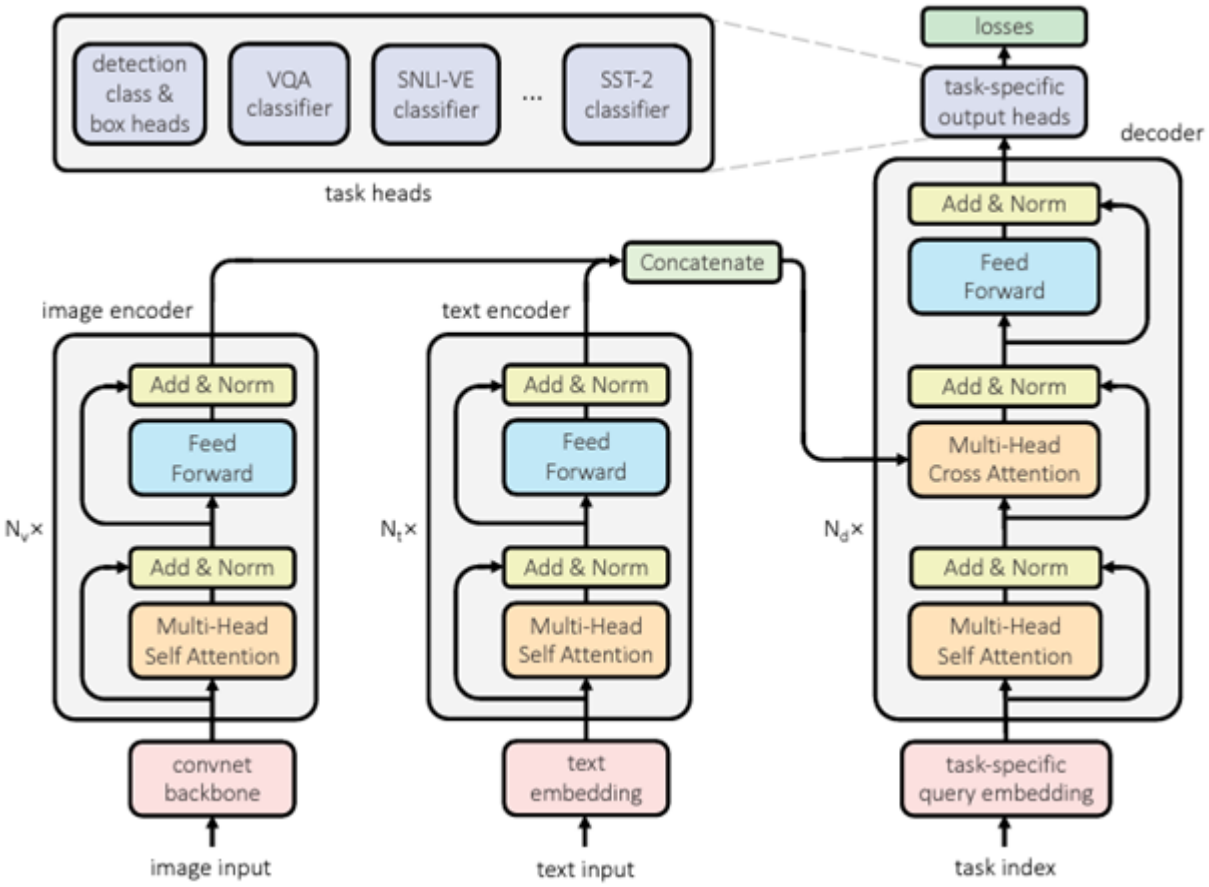
Dil, görüş ve çoklu modlu görevler üzerinde transformerlar. Transformerlar ilk olarak sequence-to-sequence modellemesi için dil alanında uygulanmıştır [1]. BERT [14], GPT [19], XLNet [20], RoBERTa [21], ALBERT [15], T5 [23], T-NLG [24] ve diğer sonraki çalışmalar, geniş metinler üzerinde önceden eğitilmiş transformerların dil temsillerini öğrenebileceğini ve bu temsillerin ince ayarlama yoluyla birçok hedef göreve aktarılabileceğini göstermiştir.

Görsel alanda, Image Transformer [25], Image GPT [26], DETR [13], ViT [12] ve diğer sonraki çalışmalar, çeşitli görsel görevler için transformer modellerini uygulamıştır. Ayrıca, transformerlardan elde edilen çoklu kafa öz-bilgi mekanizması, bir dizi görme uygulamasına da fayda sağlar (ör. [31]). Görsel soru-cevaplama gibi ortak görme-dil çıkarımı görevleri için, transformer modelleri hem görsel hem de metinsel girişleri kabul edecek şekilde genişletilmiştir (ör. VisualBERT , VILBERT [8, 15]).

Transformerların bu önceki uygulamalarının ve genişlemelerinin çoğu, ilgilenilen her bir görev için belirli bir modeli eğitmek (veya ince ayarlamak) üzerine yoğunlaşır. BERT'te [14], bir önceden eğitilmiş transformer model, birçok dil görevinde ayrı ayrı ince ayarlanır. T5'te [23], bir metin-metin transformer, farklı dil görevlerinde birlikte önceden eğitilir. Ancak, çoklu görev önceden eğitimi yoluyla genel temsiller öğrenmesine rağmen, T5 her bir hedef görev için farklı bir parametre seti üzerinde ince ayar yapar. Buna karşıt olarak, biz tek bir transformer içerisinde çoklu görevleri aynı anda öğreniyoruz.

Transformerlarla çoklu görev öğrenme. Çok modlu çoklu görev öğrenme üzerine birçok çalışma yapılmıştır, (ör. [27, 28, 30, 29, 9]). Çoklu görev öğrenmeye dair önceki çabaların çoğu, belirli alanlara veya modlara odaklanır, genellikle model mimarileri belirli bir alana uyarlanmıştır. Ancak, tek bir genel model ile farklı alanlarda çoklu görev öğrenme üzerine dikkate değer önceki çalışmalar da

vardır. [27]'de, transformer'ın çoklu kafa dikkat mekanizması temelli bir kodlayıcı-dekodlayıcı mimarinin görüntü sınıflandırma, makine çevirisi ve görüntü altyazılama gibi farklı giriş ve çıkış alanlarına uygulanabileceği gösterilmiştir. [27]'deki dekodlayıcılar, her bir çıkış görevi için özel olarak tasarlanırken, bizim modelimiz tüm görevlerde aynı dekodlayıcı mimarisini uygularken daha az görev-spesifik detay içerir. MT-DNN'de [32], bir transformer'ın alt katmanlarını paylaşarak çoklu görevli bir dil anlama modeli oluşturulurken, üst katman görev-spesifik hale getirilmiştir. VILBERT-MT'de [9], VILBERT [8] tabanlı çoklu görevli bir transformer model ile 12 görüş-dil görevi birlikte öğrenilmiştir. [32] ve [9]'a kıyasla, biz sabit giriş modlarından öteye geçerek, farklı tek modlu (yalnızca görüş ve yalnızca dil) ve çoklu modlu görevleri birleşik bir transformer modeli ile birlikte ele alıyoruz. Modelimiz, [9]'daki önceden eğitilmiş dedektörlere bağımlı olmak yerine, görüntü pikselleri üzerinde doğrudan uçtan uca eğitime izin verir.



Şekil 2.1: UniT modeline genel bakış [2]

UniT çalışmasında [2], birleştirilmiş tek bir modelle farklı modalitelerdeki birden fazla görev birlikte öğreniliyor. UniT [2], her bir giriş modalitesi türü için ayrı kodlayıcılardan ve ardından basit göreve özgü kafalara sahip bir kod çözücüden (görev başına veya paylaşılan) oluşan dönüştürücü kodlayıcı-kod çözücü mimarisi [1, 13] üzerine inşa edilmiştir. Şekil 2.1, UniT modeline bir genel bakışı temsil eder, bu model farklı alanlardaki geniş bir görev yelpazesini birleşik bir transformer kod çözücü-mimarisi ile birlikte ele alır. Model, görsel girişleri kodlamak için bir görüntü kodlayıcı kullanır, dil girişlerini kodlamak için bir metin kodlayıcısı kullanır, ve her bir görev için nihai çıktıları yapmak üzere görev-spesifik başlıklarla takip edilen her bir görev sorgusu gömme ile birlikte bir ortak dekodlayıcı kullanır.

Çoklu modlu önceden eğitim ile karşılaştırma. Voken [7] ve VisualBERT [14] gibi önceki çalışmalar, görsel altyazılar gibi çoklu modlu veriler üzerinde önceden eğitim yapmanın, görme, dil veya çoklu modlu görevlere yardımcı olduğunu gösterir, bu genellikle her bir hedef görev üzerinde ince ayarlama yaparak özelleştirilmiş modeller oluşturulması yoluyla başarılır. Bu yaklaşımlardan farklı olarak, biz tüm görevleri paylaşılan bir modelde ele alıyoruz, burada genel bilgi, belirli hedef görevlere ince ayarlama yapılması nedeniyle kaybolmaz. Farklı alanlarda çeşitli görevleri birlikte çözebilme yeteneğinin, genel zeka yolunda kritik bir adım olduğuna inanıyoruz.

3. YÖNTEM VE TEKNOLOJİLER

3.1 YÖNTEMLERİ VE KAVRAMLARI ANLAMAK

3.1.1 ÇOK MODLU MAKİNE ÖĞRENMESİ

İnsanların beş temel duyusu vardır: işitme, dokunma, koku, tat ve görme. Bu beş modaliteye sahip olduğumuzda, etrafımızdaki dünyayı algılayabilir ve anlayabiliriz. Bu nedenle, "çoklu modlu" ifadesi, çevremizi anlamak için farklı bilgi kanallarını aynı anda birleştirmeyi ifade eder. Örneğin, çocuklar "kedi" kelimesini öğrenirken, yüksek sesle kelimeyi söyleme, kedilere işaret etme ve "miyav" gibi sesler çıkarma gibi farklı modları kullanırlar. İnsan öğrenme sürecini bir rol model olarak kullanan yapay zekâ (AI) araştırmacıları da derin öğrenme modellerini eğitmek için farklı modları birleştirmeye çalışır. Yüzeysel bir seviyede, derin öğrenme algoritmaları, matematiksel olarak belirli bir kayıp fonksiyonu ile tanımlanan bir hedefi optimize etmek için eğitilmiş bir sinir ağına dayanır. Optimizasyon, yani kaybı en aza indirme, gradyan iniş adı verilen bir sayısal işlemle yapılır. Sonuç olarak, derin öğrenme modelleri sadece sayısal girişi işleyebilir ve sadece sayısal bir çıktı verebilir. Ancak, çoklu modlu görevlerde genellikle resimler veya metinler gibi yapılandırılmamış verilerle karşılaşırız. Bu yüzden, ilk büyük sorun girişi nasıl sayısal olarak temsil edeceğimizdir. Çoklu modlu görevlerle ilgili ikinci sorun, farklı modları nasıl tam olarak birleştireceğimizdir. Örneğin, tipik bir görev, bir derin öğrenme modelini bir kedi resmi oluşturacak şekilde eğitmek olabilir. Öncelikle, bilgisayarın "kedi" metin girişini anlaması ve bu bilgiyi bir şekilde belirli bir resme çevirmesi gerekir. Bu nedenle, metin girişindeki kelimeler arasındaki bağlamsal ilişkileri ve görüntü çıktısındaki pikseller arasındaki mekânsal ilişkileri belirlemek gereklidir. Okul öncesi bir çocuk için kolay olabilecek bu durum, bilgisayar için büyük bir zorluktur. Her ikisinin de "kedi" kelimesinin anlamını ve hayvanın görünümünü içeren bazı anlayışları öğrenmesi gerekir. Modern derin öğrenmedeki yaygın bir yaklaşım, kedinin sayısal olarak bir vektör olarak bazı gizli alanı temsil eden gömüler oluşturmaktır. Ancak, bunu başarmak için, son yıllarda farklı yaklaşımlar ve algoritmik mimariler geliştirilmiştir.

3.1.2 **MODALİTELER (NLP VE CV)**

Doğal Dil İşleme (NLP), yaklaşık 50 yıldır var olsa da hiç olmadığı kadar önemlidir. Bu makine öğrenmesi dalında, konuşma ve yazılı dil ile ilgilenen birçok ilerleme olmuştur. Örneğin, kelimelerin içsel gösterimlerini öğrenme, son on yılın en büyük ilerlemelerinden biriydi. Kelime gömme teknikleri (Word Embeddings), geliştiricilere kelimeleri, altta yatan semantik içeriklerini yakalayan yoğun vektörler olarak kodlama imkânı vermiştir. Bu sayede, benzer kelimeler daha düşük boyutlu bir özellik uzayında birbirlerine yakın gömülür. Encoder-decoder (yani sequence-to-sequence) mimarileri [32] bir başka önemli sorunu çözdü ve farklı uzunluklarda giriş dizilerini çıktı dizilerine haritalama imkânı sağladı. Bu mimariler, makine çevirisi, video altyazı oluşturma veya soru cevaplama gibi karmaşık görevler için özellikle yararlıdır. Bu yaklaşım, dizinin yapısı üzerinde minimal varsayımlar yapar ve farklı kelime sıraları ile aktif ve pasif sesle başa çıkabilir.

Son tekniklerden biri olan Dikkat Mekanizması [33], modellerin odaklarını aktif olarak değiştirmelerine olanak sağlar, tıpkı insanların yaptığı gibi. Bu, bir seferde bir düşünceyi takip etmeyi ve görevle ilgisiz bilgileri baskılamayı sağlar. Sonuç olarak, makine çevirisi gibi görevlerde performansın önemli ölçüde arttığı gösterilmiştir. Decoder'ın kaynağa doğrudan bakmasına izin vererek, darboğazın önüne geçilir ve aynı zamanda uzak durumları hızlandırır ve böylece kaybolan gradyan problemiyle başa çıkar. En son dizi veri modelleme tekniklerinden biri olan Transformer'lar [1], sadece dikkate dayalıdır ve giriş verilerini sıralı olarak işlemek zorunda değildir (RNN'lerin aksine). Bu nedenle, derin öğrenme modeli, uzun dizilerde daha önce bağlamla oluşturulan hatıraları daha iyi hatırlamaktadır. Şu anda NLP'deki dominant paradigma bu olup, GPU'ları daha iyi kullanabilir çünkü paralel işlemler yapabilir. Transformer mimarileri BERT [14], T5 [8] veya GPT-3 [19] gibi, büyük bir corpus üzerinde önceden eğitilmiştir ve belirli dil görevleri için ince ayar yapılabilir. Hikayeler, şiirler, kodlar ve daha fazlasını oluşturma yeteneğine sahiptirler. Yukarıda bahsedilen ilerlemelerle, derin ağlar bilgi almayı ve metin modunda semantikleri temsil etmeyi başarmıştır.

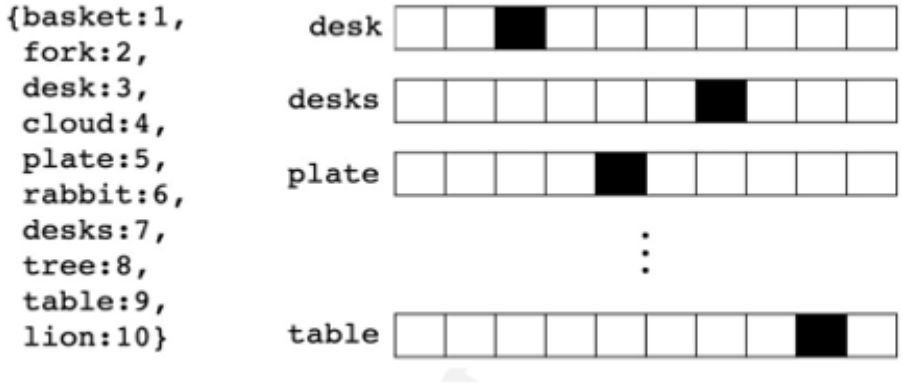
Bilgisayar görüşü (CV), insan görsel sisteminin karmaşıklığının bir kısmını taklit etmeye ve bilgisayarların nesneleri görüntülerde ve videolarda insanların yaptığı gibi tanımlamasını ve işlemesini sağlamaya odaklanır. Son yıllarda bilgisayar biliminin ana ve geniş uygulamalı alanlarından biri haline gelmiştir. Ancak, hala araştırma konusu olan sorunlar var, bu sorunların çözümleri araştırmanın konuya bakışına bağlıdır. Sorunlardan biri, görüntü sınıflandırması için derin evrişimli sinir ağlarını nasıl optimize etmektir. Sınıflandırma doğruluğu, genişlik, derinlik ve görüntü çözünürlüğüne bağlıdır. Eğitim doğruluğunun bozulmasını ele almanın bir yolu, daha iyi doğruluk elde etmek için ConvNet'leri ölçeklendirmektir, bu daha yüksek görüntü çözünürlüğüyle yapılır. Bu gözlemden yola çıkarak, basit ama etkili bir bileşik ölçeklendirme yöntemi olan EfficientNets [34] önerilmiştir.

Bilgisayar görüşünde bir başka son teknoloji trendi, insan gözetimi olmadan etkili görsel temsiller öğrenmektir. Gizli uzayda karşılaştırmalı öğrenmeye dayalı ayrımcı yaklaşımlar, son zamanlarda büyük umutlar doğurmuştur, son teknoloji sonuçlar elde etmiştir, ancak görsel temsillerin karşılaştırmalı öğrenmesi için basit bir çerçeve olan SimCLR, önceki çalışmaları geride bırakır [35]. Ancak, başka bir araştırma, alternatif olarak basit bir "takas" tahmin problemi önerir, burada bir görünümün kodunu diğer bir görünümün temsilinden tahmin ederiz.

Son yıllarda NLP ve CV'deki hızlı gelişmeyle, çok modlu görevleri ele almak için her iki modaliteyi birleştirmek sadece zaman meselesiydi. DALL-E 2'nin piyasaya sürülmesi, gelecekte bu birleşimden ne bekleyebileceğimiz konusunda bir ipucu veriyor. DALL-E 2, verilen herhangi bir metin girişinden fotoğerçekçi görüntüler veya hatta sanat eserleri oluşturabilme yeteneğine sahiptir. Yani, bir modalitenin bilgilerini başka bir modaliteye dönüştürür. Bunu mümkün kılmak için çok modlu veri setlerine ihtiyaç vardır, ki bunlar hala nispeten nadirdir. Bu, mevcut verilerin önemini ve onları daha fazla kullanma yeteneğini gösterir. Yine de, tüm modalitelerin modellerini önceden eğitmek için büyük veri setlerine ihtiyaç vardır, bu da bu tür teknolojilerin kullanımını sınırlar. Daha büyük veri setleri ve daha iyi donanım elde etme yeteneği, bu teknolojileri daha erişilebilir hale getirecektir.

3.1.3 **WORD EMBEDDİNGS (KELİME İÇ TEMSİLLERİ)**

3.1.2'de belirttiğimiz gibi, NLP'deki erken ilerlemelerden biri kelime iç temsillerini öğrenmektir. Bundan önce, metin modellemesi ile ilgili büyük bir sorun dağınıklığıydı (karışıklığıydı), çünkü makine öğrenme algoritmaları kesinlikle yapılandırılmış ve iyi tanımlanmış sabit uzunluklu girdileri tercih eder. Detaylı bir seviyede, modeller daha çok sayısal verilerle çalışmayı tercih eder. Bu yüzden, bir one-hot kodlama veya kelime çantası gibi çok basit teknikler kullanarak, bir metin bilgi kaybı olmadan sayılarının eşdeğer vektörüne dönüştürülür.



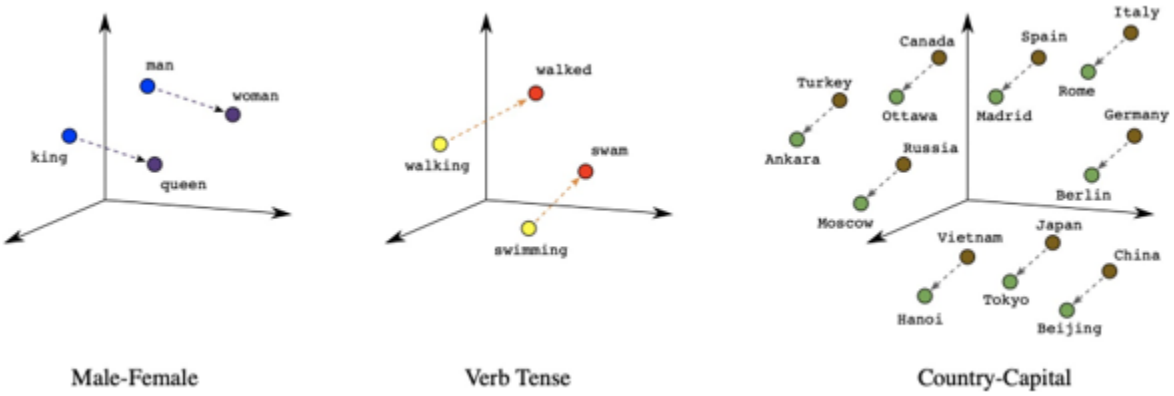
Şekil 3.1: On tane one-hot kodlanmış kelime [36]

One-hot kodlama örneğinde (Şekil 3.1), on basit kelime vardır ve koyu kareler sıfırdan farklı tek bir değere işaret eder. Buna karşılık, kelime çantası kullanılırken birden fazla sıfır olmayan değer vardır, bu da metinden özellikleri çıkarıp modellemeye kullanmak için başka bir yoldur. Bilinen kelimelerden oluşan bir kelime dağarcığından bir kelimenin mevcut olup olmadığını ölçeriz. Sıra burada dikkate alınmadığı için kelime çantası adını alır.

aşar. Klasik kelime temsillerinin diğer dezavantajları, boyut lanetinin ve genelleştirme problemi tarafından açıklanmaktadır. İlki, sözlük büyüdükçe özellik boyutunun artması nedeniyle bir problem haline gelir. Bu, seyrek ve yüksek boyutlu vektörlerle sonuçlanır. İkincisi, kelimeler arasındaki benzerlik yakalanmadığı için ortaya çıkar. Bu nedenle, daha önce öğrenilen bilgiler kullanılamaz. Ayrıca, her bir kelimeye ayrı bir vektör atama, büyük sözlükleri ve birçok nadir kelime olan diller için özellikle belirgin bir sınırlamadır.

Basit kelime temsillerinin dezavantajları ile mücadele etmek için, kelime gömme, benzer kelimelerin benzer bir kodlamaya sahip olacağı şekilde verimli ve yoğun temsilleri kullanmayı sağlar. Yani vektör uzayında daha yakın olan kelimelerin anlamda benzer olması beklenir. Gömme, kayan nokta değerlerinden oluşan bir vektör olarak tanımlanır (vektörün uzunluğu bir hiper parametredir). Gömme için değerler, bir modelin yoğun bir katman için ağırlıkları öğrenme şeklinde öğrenilen eğitilebilir parametrelerdir. Kelime temsillerinin boyutluluğu genellikle sözlükteki kelime sayısından çok daha küçüktür. Örneğin, [37] Mikolov, birkaç yüz milyon kelime için 50-100 arasında boyutların yeterli olduğunu belirtmiştir. Küçük veri setleri için kelime vektörleri için boyutluluğu 8'den başlayabilir ve daha büyük veri setleri için 1024'e kadar çıkabilir. Yeterli veri öğrenildiğinde, daha yüksek boyutların kelimeler arasındaki karmaşık ilişkileri daha iyi yakalayabileceği beklenmektedir.

Herhangi bir NLP görevi için, önceden modelinize bilgi eklemenizi kolaylaştıran kelime gömme ile başlamak mantıklıdır ve bu temel bir transfer öğrenme şekli olarak görülebilir. Nitekim bizde bu projede kelime gömmeleri kullanacağız. Gömme, kelimelerin anlamlarını temsil etmeye çalıştığı ve bunu bir dereceye kadar başardığı not edilmelidir, bir kelimenin belirli bir bağlamdaki semantiği yakalanamaz. Bu, geleneksel gömme tekniklerinde kelimelerin statik önceden hesaplanmış temsillerine sahip olması nedeniyle olur. Bu nedenle, "banka" kelimesi hem finansal bir kurumu hem de bir nehir bankasını ifade edebilir.



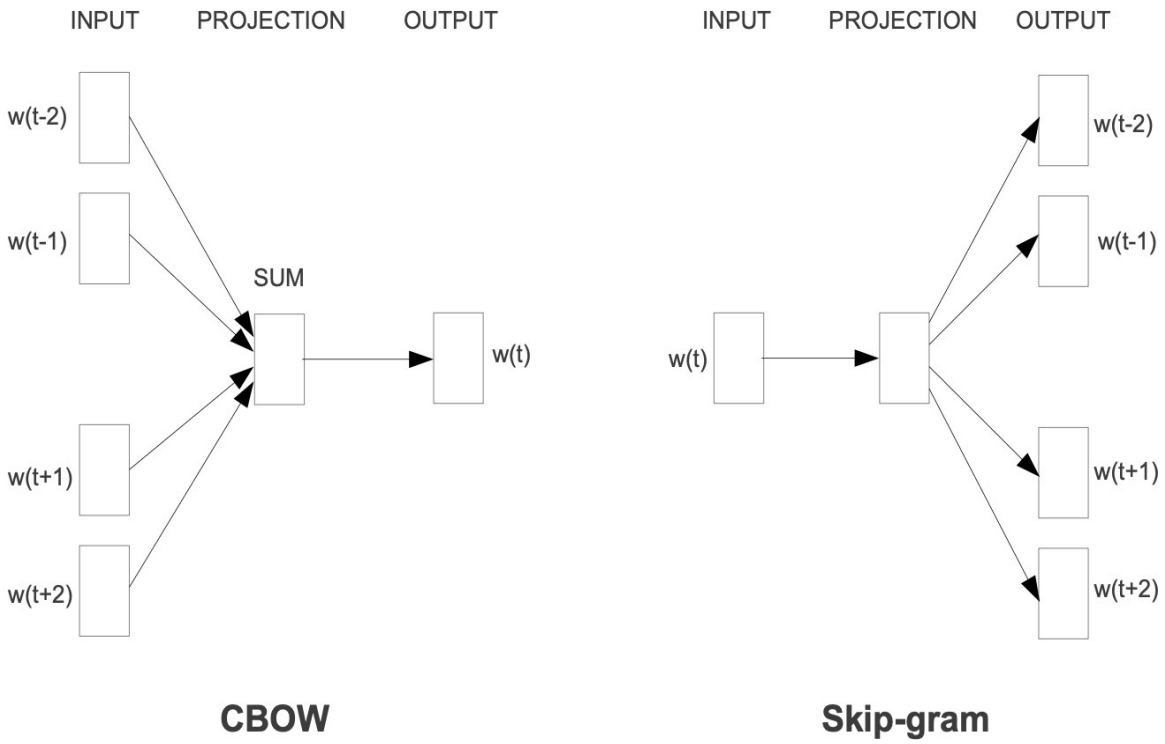
Şekil 3.2: Kelime gömme olarak üç tür benzerlik [38]

Kelimelerin çeşitli benzerlik dereceleri olabileceği not edilmelidir. Çekimli diller bağlamında, bu açık hale gelir çünkü kelimeler gramer kategorilerini ifade etmek için ayarlanır. Örneğin, orijinal vektörün bir alt uzayında, benzer sonlara sahip isimler bulunabilir. Ancak, bu basit sözdizimsel düzenlilikleri bile aşar. Kelime vektörleri üzerinde basit işlemlerle, vektör(King) - vektör(Man) + vektör(Woman)'ın anlamda "Queen" kelimesine en yakın olan bir vektöre eşit olduğu gösterilebilir. Bu ilişkinin basit bir görselleştirilmesi aşağıdaki sol grafikte görülebilir (Şekil 3.2'e bakınız). Üç koordinat sistemi, boyut azaltma teknikleri aracılığıyla bu şekilde betimlenen daha yüksek boyutların temsilleridir. Ayrıca, fiil-zaman ilişkisi orta grafikte ifade edilir, bu da kelime sonlarının benzer olduğuna dair önceki bilgiden genişletir çünkü bu durumda yürüme ve yüzme fiillerinin geçmiş zamanları yapısal olarak benzer değildir. Ayrıca, figürün sağ tarafında, [37] Mikolov tarafından verilen Ülke-Başkent örneği olarak adlandırılan ve kolayca anlaşılan bir form bulunmaktadır.

Şimdi dikkatimizi en etkili gömme tekniklerinden biri olan word2vec'e çevirelim. Mikolov [37] tarafından önerilmiştir ve tekil bir algoritma değildir. Bunun yerine, kelime temsillerini öğrenmek için bir dizi model mimarisi ve optimizasyon olarak görülebilir. Word2vec'in popülaritesi, birden çok doğal dil işleme görevindeki başarısından da kaynaklanmaktadır.

Word2vec, temel bir ileri beslemeli sinir ağı üzerine kurulu çok basit bir yapıya sahiptir. Mikolov [37] (2013a, 2013c, 2013d), kelime gömme öğrenme konusunda iki farklı ancak ilgili yöntem üzerine odaklanan bir dizi makale yayınladılar (Şekil 3.3'e bakınız). Birinci olarak Sürekli kelime çantası modeli,

çevreleyen bağlam kelimelerine dayanarak ortadaki kelimeyi tahmin etmeyi amaçlar. Bu nedenle, hedef kelimenin öncesi ve sonrasındaki bileşenleri dikkate alır. Bağlamdaki kelimelerin sırası önemli olmadığından, bu model kelime çantası modeli olarak adlandırılır. İkinci olarak, Sürekli skip-gram modeli sadece mevcut kelimeyi dikkate alır ve aynı cümledeki diğer kelimeleri ondan önce ve sonra belirli bir aralıkta tahmin eder. Her iki model de çıktı katmanı için softmax sınıflandırıcısını kullanır.

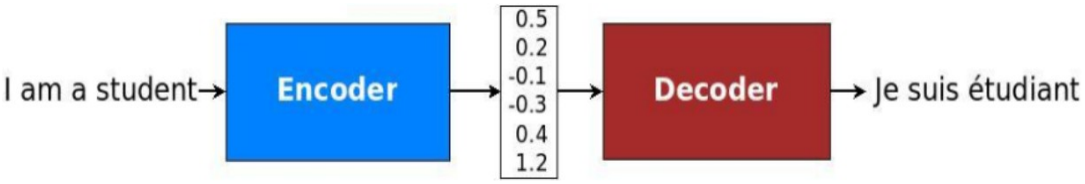


Şekil 3.3: CBOW ve Skip-gram mimarisi [37]

Daha sonra Bojanowski [39], kelimelerin morfolojisini (iç yapısını) hesaba katarak skip-gram modellerini genişletti. En azından ismi anılması gereken farklı bir klasik gömme mimarisi ise GloVe modelidir. Bu model, bir sinir ağı kullanmaz, ancak yerel bağlam bilgilerini küresel eş oluşum istatistikleriyle birleştirir.

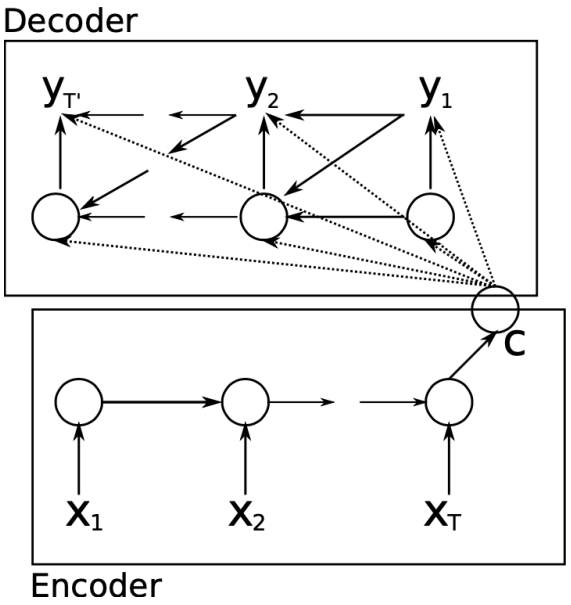
3.1.4 **KODLAYICI ÇÖZÜCÜ (ENCODER DECODER)**

Doğal dil işleme alanı, metinle ilgili çeşitli görevlerle ilgilenir. NLP probleminin türüne bağlı olarak, ağ, girdi ve/veya çıktı olarak değişken uzunlukta dizilerle karşılaşabilir. Bu durum, soru cevaplama, diyalog sistemleri veya makine çevirisi gibi birçok zorlayıcı uygulama için geçerlidir. Aşağıda, çeviri görevleri söz konusu olduğunda, giriş dizilerinin farklı uzunluklarda çıkış dizilerine eşlenmesi gerektiği açık halde gösterilmiştir. Bu tür giriş ve çıkışları yönetmek için, iki ana bölümlü bir tasarım yararlı olabilir. İlk kısma kodlayıcı denir çünkü ağın bu kısmında, değişken uzunlukta bir giriş dizisi sabit bir duruma dönüştürülür. Sonraki kısım olan çözücü, kodlanmış durumu değişken uzunlukta bir çıkış dizisine eşler. Bütün olarak, bu yapıya kodlayıcı-çözücü veya dizi-dizi mimarisi denir ve geçitli gizli birimlere (gated hidden units) sahip tekrarlayan sinir ağlarının bile başarıyla çözmekte zorlandığı birçok uygulama için etkili ve standart bir yaklaşım haline gelmiştir. Derin RNN'ler bir şansı olabilir, ancak kodlayıcı-çözücü gibi farklı mimarilerin en etkili olduğu kanıtlanmıştır. Hatta farklı kelime sıralarıyla ve aktif-pasif sesle bile başa çıkabilir [32]. Kodlayıcı-çözücü modelinin basitleştirilmiş bir örneği şekil 3.4'de görülebilir.



Şekil 3.4: Basitleştirilmiş seq2seq modeli aracılığıyla çeviri [40]

Konseptleri niceliklendiren denklemlere geçmeden önce, Cho ve diğerleri [33] tarafından önerilen dizi-dizi tasarımını incelemek mantıklıdır. Bir kodlayıcı-RNN, n_x uzunluğundaki giriş dizisini işler ve genellikle kodlayıcının son gizli durumu veya gizli durumların basit bir işlevi olan sabit uzunluklu bir bağlam vektörü C hesaplar. Giriş dizisi işlendikten sonra gizli duruma eklenir ve kodlayıcıdaki gizli durumlar arasındaki tekrarlayan bağlantılar aracılığıyla zaman içinde ileriye doğru aktarılır. Bağlam vektörü genellikle son gizli durumun basit bir işlevi olmasına rağmen, rolü küçümsenemez. Özellikle, kodlanmış durum, giriş dizisinden önemli bilgileri özetler, örneğin, bir soru cevaplama görevindeki sorunun niyeti veya makine çevirisi durumunda bir metnin anlamı. Bağlam, çözücünün her gizli durumuna aktarıldıktan sonra, çözücü RNN bu bilgiyi, n_x 'den farklılık gösterebilecek n_y uzunluğunda hedef diziyi üretmek için kullanır.



Şekil 3.5: kodlayıcı-çözücü mimarisi [33]

Çözücü, hedefi son zaman adımındaki gizli duruma dayanarak tahmin etmek için eğitilmiş başka bir RNN türüdür. Ancak, düzenli RNN'lerden farklı olarak, aynı zamanda son zaman adımının çıktısına (y_{t-1}) ve girişin bir özetine (c) dayanır. Bu nedenle, çözücünün gizli durumu şu şekilde hesaplanır:

$$h_d^{[t]} = f\left(h_d^{[t-1]}, y^{[t-1]}, c\right)$$

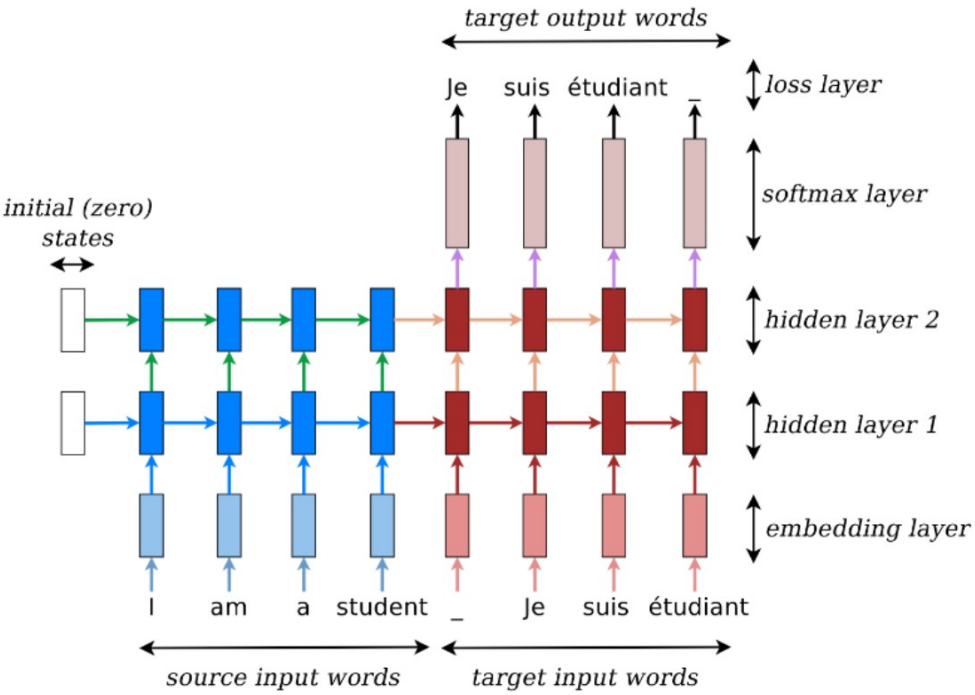
Benzer şekilde, her koşullu olasılık aşağıdaki gibidir, burada f, bir doğrusal olmayan aktivasyon fonksiyonudur (ve olasılıklar üretmelidir, örneğin softmax fonksiyonu):

$$P\left(y^{[t]} \middle| y^{[1]}, \ldots, y^{[t-1]}, C\right) = f\left(h_d^{[t-1]}, y^{[t-1]}, c\right)$$

İki bölüm, koşullu log-olasılığın maksimize edilmesi için birlikte eğitilir, burada θ model parametrelerinin setini ve (xn, yn) N boyutlu eğitim setinden bir (giriş dizi, çıkış dizi) çiftini gösterir:

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} log\ p_\theta(y_n|x_n)$$

En iyi olasılık genellikle ışın arama algoritması kullanılarak bulunur. Bunun temel fikri, çözücünün her adımında, en olası k kısmi çevirileri (hipotez olarak adlandırılır) izlemektir.



Şekil 3.6: Seq2seq modeli aracılığıyla çeviri [40].

Zaman boyunca açılan gizli birimlerle sunulan çevirinin incelenmesi, şekil 3.6'de görülebilir. Özellikle, birden fazla gizli katmanın kullanılması araştırmacılar tarafından önerilmektedir. Fikir, alt katmanların düşük seviye özellikleri ve üst katmanların yüksek seviye özellikleri hesaplamasıdır.

Geçitli tekrarlayan ağlar, özellikle uzun kısa süreli hafıza ağları (LSTM), dizi-dizi mimarisinin her iki bileşeninde de etkili bulunmuştur. Ayrıca, derin LSTM'lerin sığ LSTM'lerden önemli ölçüde daha iyi performans gösterdiği ortaya çıkmıştır. Her bir ek katman, muhtemelen çok daha büyük gizli durumları nedeniyle, karmaşıklığı yaklaşık %10 oranında azaltmıştır. Örneğin, Sutskever ve diğerleri [32], 1000 boyutlu kelime gömüleri için her katmanda 1000 hücreli 4 katmanlı derin LSTM'ler kullanmıştır. Böylece, bir cümleyi temsil etmek için toplamda 8000 gerçek sayı kullanılmıştır. Basitleştirmek için, sinir ağları aşağıda RNN'ler olarak anılmaktadır, bu da LSTM'ler bir tür geçitli RNN olduğu için bu paragrafın öngörüleriyle çelişmemektedir [32].

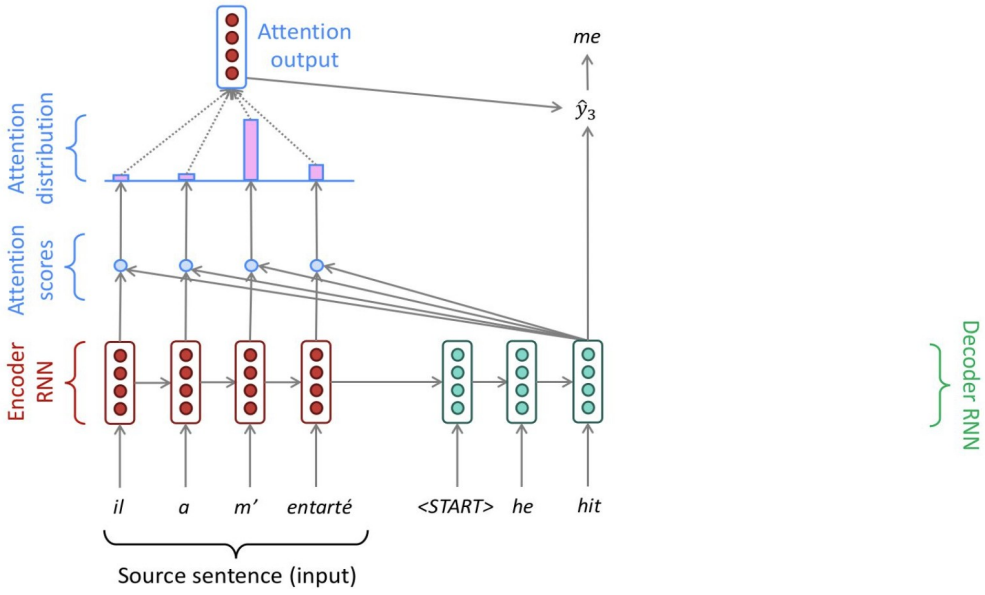
3.1.5 **DİKKAT (ATTENTİON)**

Kodlayıcı-kod çözücü mimarileri değişken uzunluktaki dizilerle uğraşmayı kolaylaştırmış olsa da aynı zamanda karmaşıklıklara da neden olmuştur. Tasarımları gereği, kaynak cümlenin kodlaması tek bir vektör temsilidir (bağlam vektörü). Sorun, bu durumun kaynak cümle hakkındaki tüm bilgileri tek bir vektörde sıkıştırması gerektiğidir ve bu durum genel olarak darboğaz sorunu olarak adlandırılır. Kesin olmak gerekirse, keyfi uzunluktaki cümlelerin tüm semantiği tek bir gizli duruma sıkıştırılmalıdır. Dahası, bilginin çok sayıda zaman adımı arasında aktarılması gerektiğinden farklı bir öğrenme problemi oluşturur. Bu, ağ içindeki kaybolan gradyanlara yol açar çünkü faktörlerin her biri her noktada 1'den küçük bir değerle çarpılır. Örneğin, son cümle, bir kod çözücü yaklaşımının başa çıkmakta zorlanabileceği ideal bir örnektir. Özellikle, cümleler eğitim derleminden daha uzunsa [40].

Yukarıda bahsedilen nedenlerden dolayı, Bahdanau ve diğerleri [33] tarafından hizalamayı ve çeviriyi birlikte öğrenen diziden diziye mimarisinin bir uzantısı önerilmiştir. Model, üretilen her kelime için kaynak cümledeki en alakalı bilginin bulunduğu bazı pozisyonları tarar. Daha sonra, çevresindeki bağlam ve daha önce oluşturulan kelimelere dayanarak, model mevcut zaman adımı için hedef kelimeyi tahmin eder. Bu yaklaşım, insan benzeri (bilişsel) dikkati taklit ettiği için dikkat olarak adlandırılır. Kaynağa doğrudan bakarak ve darboğazı aşarak soruna bir çözüm sunar. Ardından, kaybolan gradyan sorununu hafifletir, çünkü artık uzaktaki durumlara giden bir kestirme yol vardır. Sonuç olarak, dikkat mekanizmasının dahil edilmesinin NLP görevlerinde modellerin performansını önemli ölçüde artırdığı gösterilmiştir.

Aşağıdaki örneğe, şekil 3.7'ye göz atıldığında, dikkat mekanizmasının işleyişine ilişkin tüm soru işaretleri giderilmiş olacaktır. Sol altta Fransızca olarak verilen ve kodlayıcı-RNN (kırmızı renkte) için girdi görevi gören kaynak cümle görülmektedir. Ardından, dikkat puanları (mavi renkte) bir önceki çıkış kelimesi ile giriş kelimeleri arasındaki nokta çarpımı alınarak hesaplanır. Daha sonra, softmax işlevi puanları bir olasılık dağılımına (pembe) dönüştürür. Bunlar, kodlayıcının gizli durumlarının ağırlıklı toplamını almak ve yüksek dikkat alan gizli

durumlardan çoğunlukla bilgi içeren dikkat çıktısını oluşturmak için kullanılır. Daha sonra, dikkat çıktısı, daha önce olduğu gibi kod çözücü çıktısını hesaplamak için uygulanan kod çözücü gizli durumu (yeşil renkte) ile birleştirilir. Bazı durumlarda, dikkat çıktısı da çözücüyü (normal çözücü girişiyle birlikte) besler. Bu özel örnek, "entarter"ın "birini pasta ile vurmak" anlamına geldiği ve bu nedenle birçok kelimeyle çevrilmesi gereken bir kelime olduğu için seçildi. Bu ifade için mevcut doğrudan eşdeğerlerin olmamasının bir sonucu olarak, neredeyse sıfır olmayan tek bir skorun olmaması (birden fazla kelime ile çevrileceğinden dolayı) beklenmektedir. Bu anlık görüntüde, dikkat dağılımının iki önemli katkısı olduğu görülebilir.



Şekil 3.7: Dikkat mekanizması ile çeviri süreci [40].

Aşağıdaki denklemler, son paragraflarda ortaya konan ilişkileri derli toplu bir şekilde temsil etmeyi amaçlamakta ve esas olarak Manning ve diğerlerinin çalışmasından yararlanmaktadır [40]. Dikkat puanları $e^{[t]}$, çözücünün gizli durumunun kodlayıcının tüm gizli durumlarıyla skaler olarak birleştirilmesiyle hesaplanır:

$$e^{[t]} = \left[\left(h_d^{[t]}\right)^T h_e^{[1]}, \ldots, \left(h_d^{[t]}\right)^T h_e^{[N]}\right]$$

Temel nokta çarpımı dikkatine (basic dot-product attention) ek olarak, dikkat puanlarını hesaplamanın başka yolları da vardır, örneğin çarpımsal veya

eklemeli dikkat aracılığıyla. Daha sonra, skaler puanlara softmax uygulandığında, değerleri toplamı 1 olan bir olasılık dağılımı olan $\alpha^{[t]}$ dikkat dağılımı elde edilir:

$$\alpha^{[t]} = softmax\left(e^{[t]}\right)$$

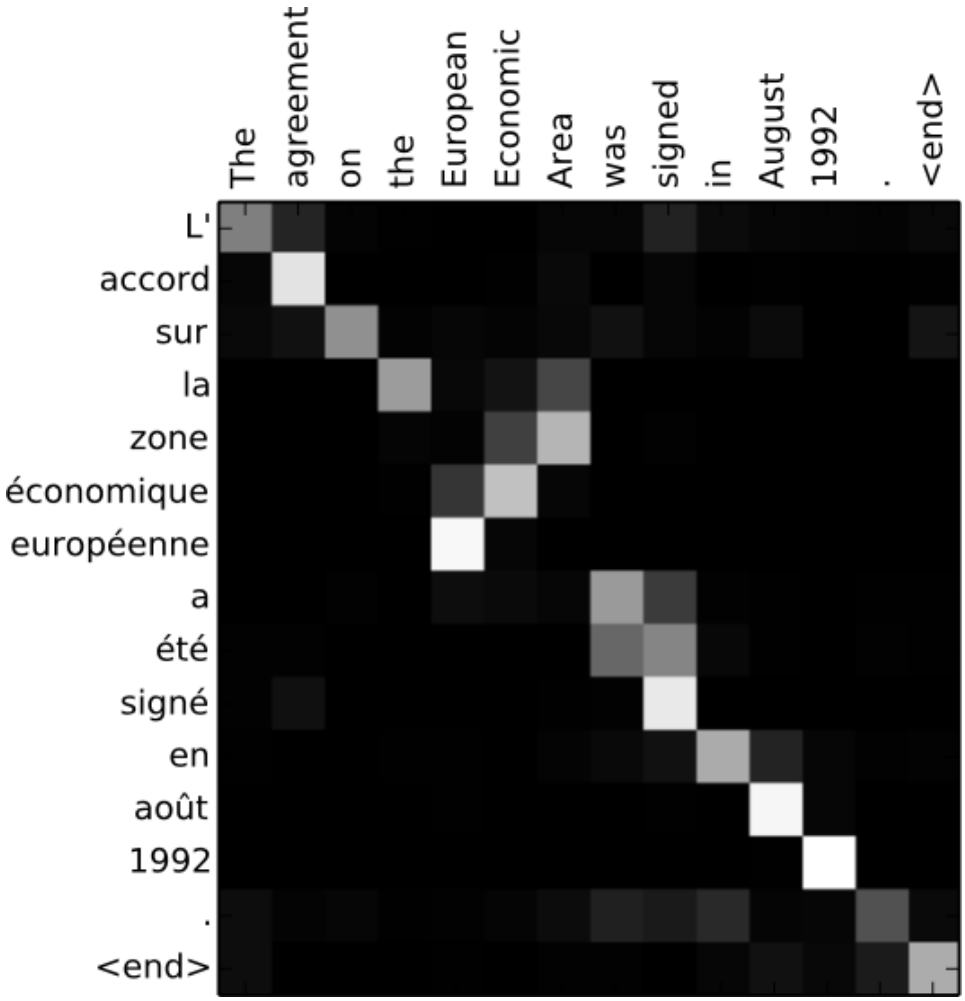
Daha sonra, dikkat çıktısı $\alpha^{[t]}$, kodlayıcının gizli durumları için bir ağırlık olarak hareket eden dikkat dağılımı ile elde edilir:

$$\alpha^{[t]} = \sum_{i=1}^{N} \alpha_i^{[t]} h_{e,i}$$

Dikkat çıktısını çözücü gizli durumuyla birleştirmek ve dikkat almayan dizi-dizi modelinde olduğu gibi sürece devam etmek son adımlardır:

$$\alpha^{[t]} = f\left(a^{[t]} h_d^{[t]}\right)$$

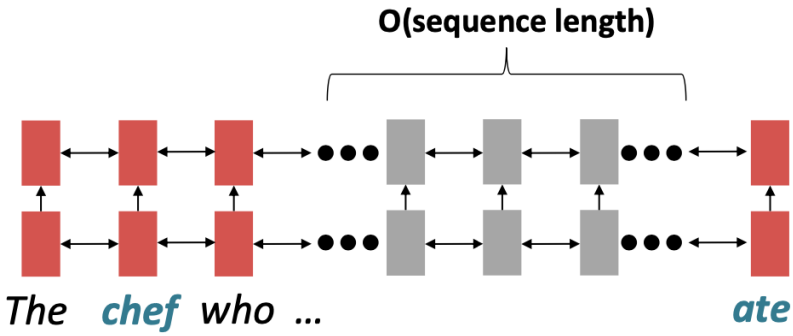
Hizalamalar olarak da adlandırılan dikkat dağılımını görselleştirerek [33] kod çözücünün neye odaklandığını gözlemlemek ve neden belirli bir çeviriyi seçtiğini anlamak kolaydır. Şekil 3.8'deki grafiğin x ekseni kaynak cümledeki (İngilizce) kelimelere, y ekseni ise oluşturulan çevirideki (Fransızca) kelimelere karşılık gelmektedir. Her piksel, 0'ın siyah ve 1'in beyaz olduğu gri tonlamalı ilgili hedef kelime için kaynak kelimenin ağırlığını gösterir. Sonuç olarak, hedef kelime oluşturulurken kaynak cümledeki hangi konumların daha alakalı olduğu ortaya çıkmaktadır. Beklendiği gibi, İngilizce ve Fransızca arasındaki hizalama büyük ölçüde monotoniktir, çünkü kelimler çoğunlukla birer eş anlamlılıkla hizalanmaktadır ve bu nedenle matrisin ana köşegeni boyunca ağırlıklar daha yüksektir. Bununla birlikte, sıfatlar ve isimler tipik olarak iki dil arasında farklı sıralandığı için bir istisna vardır. Bu nedenle, model (doğru bir şekilde) " European Economic Area"yı "zone économique européene" olarak çevirdi. İki kelime ("Avrupa" ve "Ekonomik") üzerinden atlayarak, "zone" ile "area"yı hizaladı. Daha sonra, "zone économique européene" ifadesini mükemmelleştirmek için iki kez bir kelime geriye döndü. Ek nitel analiz, model hizalamalarının çoğunlukla sezgimize benzer olduğunu göstermiştir.



Şekil 3.8: Dikkat hizalamaları [33].

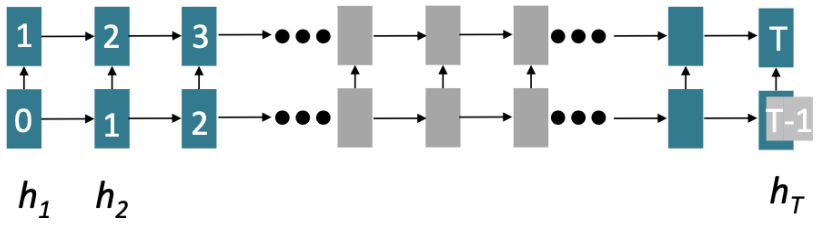
3.1.6 **DÖNÜŞTÜRÜCÜ AĞLAR (TRANSFORMER)**

RNN'ler bir taraftan diğer tarafa doğru açılır. Yani, soldan sağa ve sağdan sola. Bu, yakın kelimeler genellikle birbirlerinin anlamını etkilediği için faydalı bir sezgisel olan doğrusal yerelliği kodlar. Peki uzaktaki kelimelerin birbiriyle etkileşime geçmesi gerektiğinde durumlar nasıldır? Örneğin, bir metin bölümünün başında bir kişiye atıfta bulunur ve yalnızca en sonunda ona geri dönersek, aradaki tüm metnin geriye doğru izlenmesi gerekir şekil 3.9 ve 3.10'a bakınız. Dolayısıyla, RNN'ler uzaktaki kelime çiftlerinin etkileşim için *O*(dizi uzunluğu) adım atar. Gradyan sorunları nedeniyle, uzun mesafeli bağımlılıkları öğrenmek zordur. Buna ek olarak, doğrusal düzen yerleşiktir ve bu sıralı yapı, bilindiği gibi, tüm hikayeyi anlatmaz.



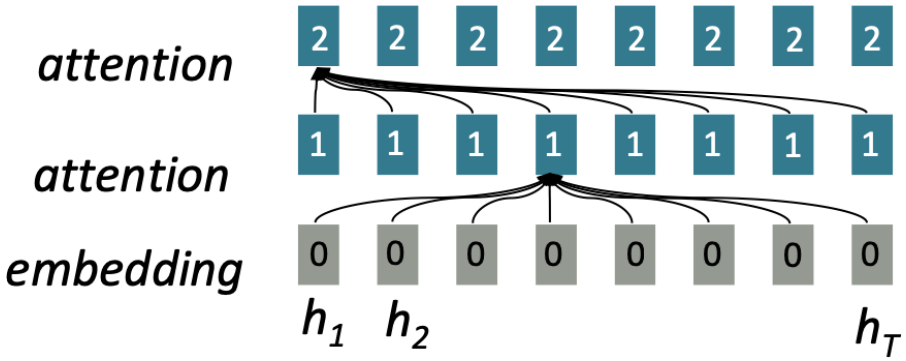
Şekil 3.9: Tekrarlayan modelin sıralı işlenmesi [40].

GPU'lar birden fazla hesaplamayı aynı anda gerçekleştirebilir ve derin öğrenme algoritmasının yürütme süresini büyük ölçüde azaltmaya yardımcı olabilir. Ancak, ileri ve geri geçişler tekrarlanan modellerde paralelleştirilemez ve *O*(dizi uzunluğu) değerine sahiptir. Daha kesin olmak gerekirse, geçmiş durumlar hesaplanmadan gelecekteki gizli durumlar tam olarak hesaplanamaz. Bu, büyük veri setleri üzerinde eğitimi engeller. İlgili durum hesaplanmadan önceki minimum adım sayısını gösterir.



Dikkatin performansı önemli ölçüde artırdığını kanıtladıktan sonra, Google araştırmacıları bunu daha da ileri götürdü ve dönüştürücüleri yalnızca dikkat üzerine, yani herhangi bir RNN olmadan temellendirdi. Bu nedenle, tanıtıldıkları makalenin adı "Attention is all you need" (İhtiyacınız olan tek şey dikkat). Spoiler: İhtiyacımız olan tek şey bu değil, ancak ilerleyen sayfalarda bu konuda daha fazla bilgi vereceğiz. Dönüştürücüler, makine çevirisi ve belge üretimi gibi birçok ortamda harika sonuçlar elde etmiştir. Paralelleştirilebilirlikleri, verimli bir şekilde önceden eğitilmelerini sağlar ve onları standart model mimarisi haline getirir. Aslında, popüler toplu benchmark olan GLUE'deki tüm en iyi modeller önceden eğitilmiş ve Dönüştürücü tabanlıdır. Ayrıca, NLP'nin dışında da örneğin Görüntü Sınıflandırma, Protein Katlama ve Sistemler için ML'de [10, 11, 41] vaat edici sonuçlar göstermişlerdir.

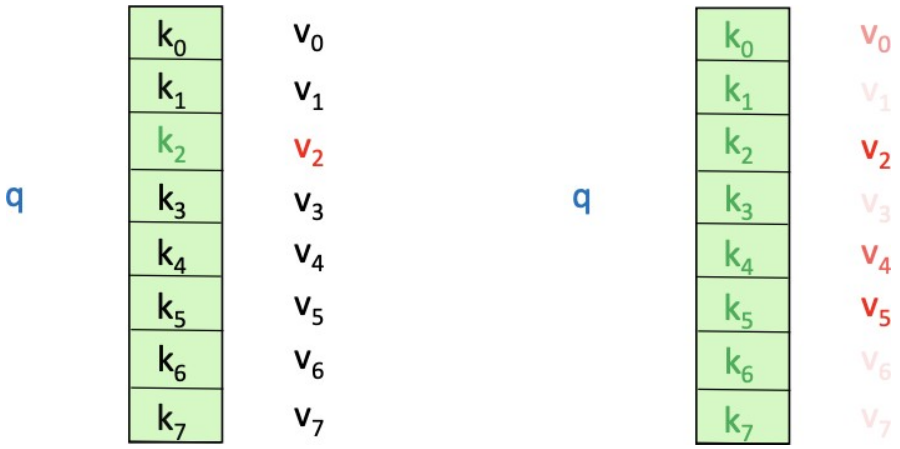
Yinelemenin kusurları varsa, dikkat mekanizmasının başka bir şekilde ayarlanması faydalı olabilir. Şimdiye kadar dikkat, kod çözücüden kodlayıcıya doğru tanımlanıyordu. Alternatif olarak, dikkat bir durumdan aynı kümedeki tüm durumlara da olabilir. Bu, kodlayıcı-kodlayıcı veya çözücü-çözücü dikkati (kodlayıcı-çözücü yerine) olan ve dönüştürücü mimarisinin temel taşını temsil eden öz dikkatin tanımıdır. Her kelimenin bir önceki katmandaki tüm kelimelere dikkat ettiği bu süreci tasvir eder. Pratikte çoğu satır eninde sonunda atlansa da.



Şekil 3.11: Klasik dikkat mekanizmasının bağlantıları [40].

Öz-dikkati yaklaşık bir hash tablosu olarak düşünmek, sezgisini anlamayı kolaylaştırır. Bir değeri aramak için, sorgular bir tablodaki anahtarlarla karşılaştırılır. Şekil 3.12'nin sol tarafında gösterilen bir hash tablosunda, her sorgu için tam olarak bir anahtar-değer çifti vardır. Buna karşılık, öz-dikkatte, her anahtar

her sorgu tarafından değişen derecelerde eşleştirilir. Böylece, sorgu-anahtar eşleşmesine göre ağırlıklandırılmış bir değerler toplamı döndürülür.



Şekil 3.12: Klasik dikkat mekanizmasının hash tabloları ile öz dikkat ile karşılaştırılması [40].

Son paragrafta kısaca tanımlanan süreç, aşağıdaki adımları takip eder ve Manning ve diğerleri [40] tarafından özetlenebilir. İlk olarak, her x_i kelimesi için sorgu, anahtar ve değer türetmek gereklidir:

$$q_i = w^Q x_i, \qquad k_i = w^K x_i, \qquad v_i = w^V x_i$$

İkinci olarak, dikkat skorları hesaplanmalıdır:

$$e_{ij} = q_i k_j$$

Üçüncü olarak, dikkat skorlarını normalize etmek için softmax fonksiyonu uygulanır:

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_j e_{ij}}$$

Son olarak, değerlerin ağırlıklı toplamını almak dikkat çıktısının elde edilmesini sağlar:

$$\alpha_i = \sum_j \alpha_{ij} v_j$$

Yinelemeler yerine öz-dikkatin dahil edilmesinin birçok avantajı ortaya çıkmıştır. Tüm kelimeler her katmanda etkileşime girdiğinden, maksimum etkileşim mesafesi O(1)'dir ve bu çok önemli bir yükseltmedir. Buna ek olarak,

model derinlemesine çift yönlüdür çünkü her kelime bağlama her iki yönde de katılır. Bu ilerlemelerin bir sonucu olarak, katman başına tüm kelime temsilleri paralel olarak hesaplanabilir. Yine de bazı konuların tartışılması gerekmektedir. Yani sinir ağları olmadan, eleman bazında doğrusal olmayan özellikler yoktur. Bunların önemi küçümsenemez ve dikkatin neden aslında ihtiyaç duyulan tek şey olmadığını gösterir. Ayrıca, çift yönlülük her zaman istenen bir durum değildir. Dil modellemesinde, modelin sadece ileriye bakmasına özellikle izin verilmemeli ve hedefin izin verdiğinden daha fazlasını gözlemlemelidir. Dahası, kelime sırası artık kodlayıcı değildir ve bir kez daha kelime torbasıdır.

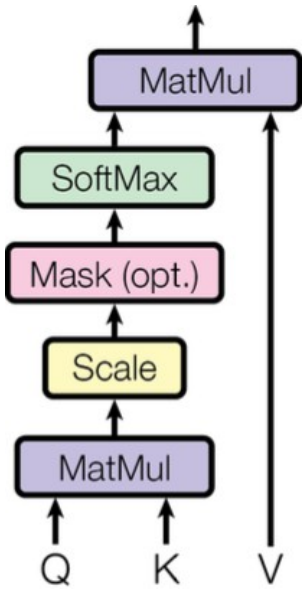
Neyse ki, Vaswani ve diğerleri [1] tarafından önerilen orijinal dönüştürücü mimarisi için daha önce bahsedilen zayıflıklar ele alınmıştır. İlk sorun, dikkat çıktısına bir ileri besleme katmanı uygulanarak kolayca çözülebilir. Bu, doğrusal olmayan aktivasyonun yanı sıra ekstra ifade gücü de sağlar. Ardından, çift yönlülüğün öğrenme hedefiyle çeliştiği durumlar için, gelecekteki durumlar maskelenebilir, böylece dikkat önceki durumlarla sınırlandırılabilir. Dahası, kelime kaybı, girdilere konum temsilleri eklenerek düzeltilebilir.

Derin öğrenme modelleri ne kadar karmaşık olursa, gerçek dünyanın karmaşıklığını modellemeye o kadar yakın olurlar. Bu nedenle dönüştürücü kodlayıcı ve kod çözücü, cümlelerden hem sözdizimsel hem de anlamsal özellikleri çıkarmak için gerekli olan ileri beslemeli bir ağ ile birçok öz dikkat katmanından oluşur. Aksi takdirde, kelimeler arasında anlamsal olarak derin temsiller olan kelime gömülerini kullanmak gereksiz olacaktır (Sejnowski, 2020). Aynı zamanda, derin ağları eğitmek zahmetli olabilir. Bu nedenle, eğitim sürecine yardımcı olmak için bazı hileler uygulanır.

Bunlardan biri, "ham" gömmeleri (embeddings) doğrudan bir sonraki katmana aktarmaktır, bu da birçok katmandan geçerken önemli bilgilerin unutulmasını veya yanlış temsil edilmesini önler. Bu işleme artık bağlantılar adı verilir (residual connections) ve çerçeve kaybını (the loss landscape) düzelttiğine

inanılır. Ek olarak, girdileri alttaki katmanlar nedeniyle değişmeye devam ettiğinde belirli bir katmanın parametrelerini eğitmek sorunludur.

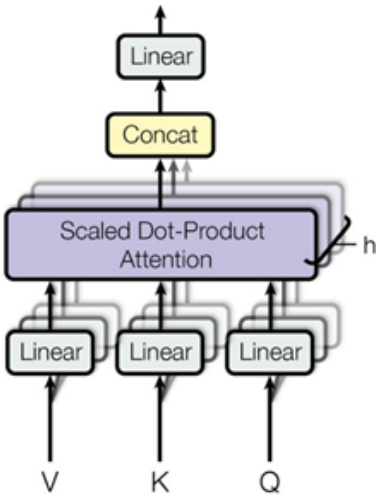
Her katmanda ortalamayı sıfıra ve standart sapmayı bire normalleştirerek bilgilendirici olmayan varyasyonu azaltmak bu etkiyi zayıflatır. Bir başka zorluk da nokta çarpımının artan boyutluluk d_k ile ölçeklenen varyans nedeniyle uç değerler alma eğiliminden kaynaklanmaktadır. Bu sorun, sorgunun anahtarlarıyla nokta çarpımlarının hesaplanması, anahtarların $\sqrt{d_k}$ boyutuna bölünmesi ve değerlerin ağırlıklarını almak için bir sonraki softmax fonksiyonunun uygulanmasından oluşan Ölçeklendirilmiş Nokta Çarpımı Dikkati (bkz. Şekil 3.13) ile çözülür.



Şekil 3.13: Ölçeklendirilmiş nokta çarpımı dikkati [1].

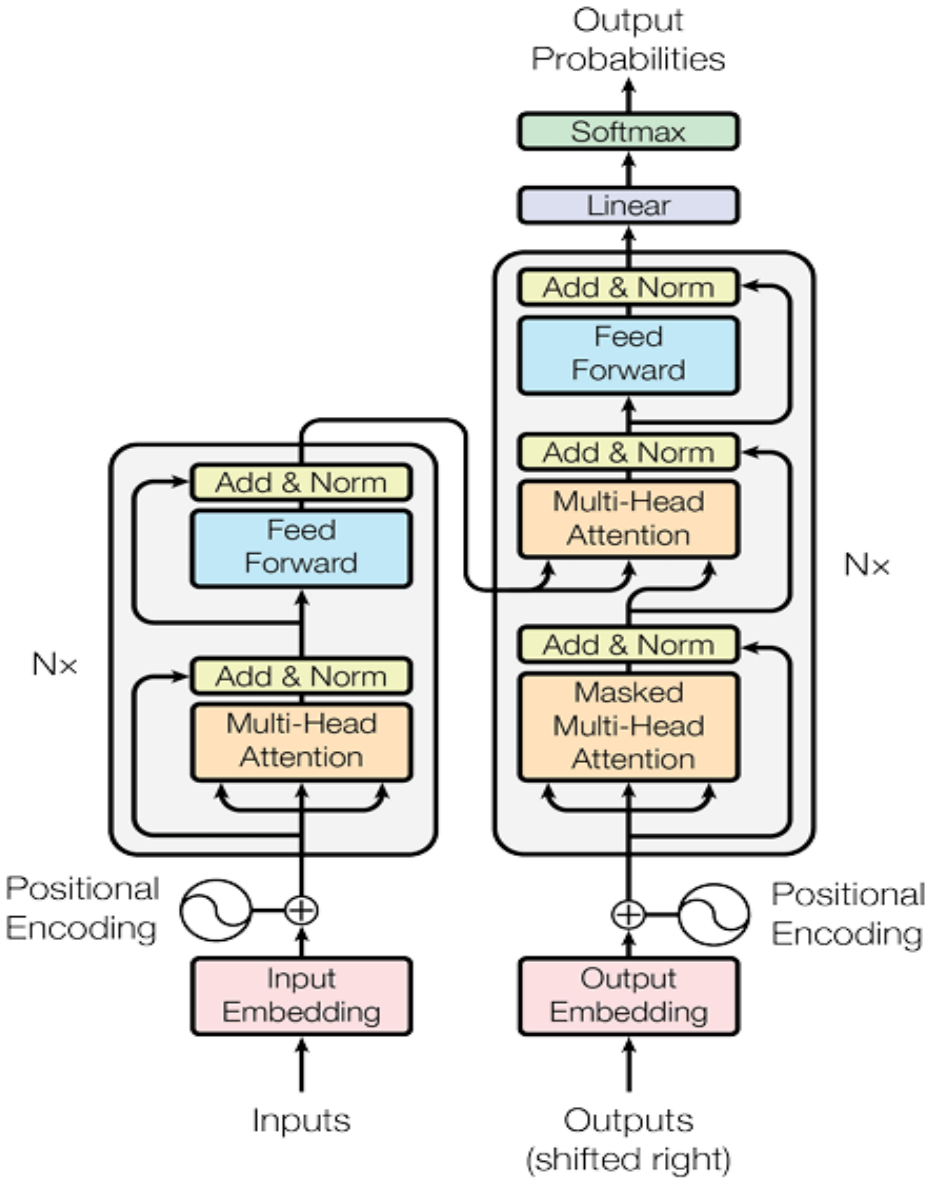
Dikkat, ilgili bilgiyi nerede arayacağını öğrenir. Şüphesiz, bir cümledeki farklı bilgi türlerine aynı anda dikkat etmek daha da umut verici sonuçlar verir. Bunu uygulamak için fikir, katman başına birden fazla dikkat kafasına sahip olmaktır. Bir dikkat kafası yoğun bilgilere dikkat etmeyi öğrenirken, bir diğeri ilgili konulara dikkat etmeyi öğrenebilir. Böylece, her kafa ayrı özelliklere odaklanır ve değer vektörlerini farklı şekilde oluşturur. Çok başlı öz dikkat (bkz. Şekil 3.14), basitçe n adet bağımsız dikkat mekanizması oluşturup bunların çıktılarını

birleştirerek uygulanır.



Şekil 3.14: Çok başlı dikkat [1].

Bu noktada, dönüştürücü mimarisinde kodlayıcıyı oluşturan her parça tanıtılmıştır (bkz. Şekil 3.15 veya Şekil 1.1). İlk olarak, konumsal kodlamalar girdi gömülmelerine dahil edilir. Bu adımı gerçekleştirmek için birden fazla seçenek vardır, örneğin sinüzoitler (sinusoids) aracılığıyla. Az önce bahsedilen çoklu kafa dikkati bunu destekler. "Add & Norm" artık bağlantılar (residual connections) ve normalleştirme katmanı (layer normalization) anlamına gelir. Bunu, artık bağlantılar ve normalleştirme katmanının da eşlik ettiği ileri beslemeli bir ağ takip eder. Tüm bunlar n kez tekrarlanır. Kod çözücü için, bireysel bileşenler benzerdir. Bir fark, çıktıların çok kafalı dikkat ve ileri besleme ağından (artık bağlantılar ve katman normalizasyonu ile) önce maskelenmiş çok kafalı dikkatten geçmesidir.

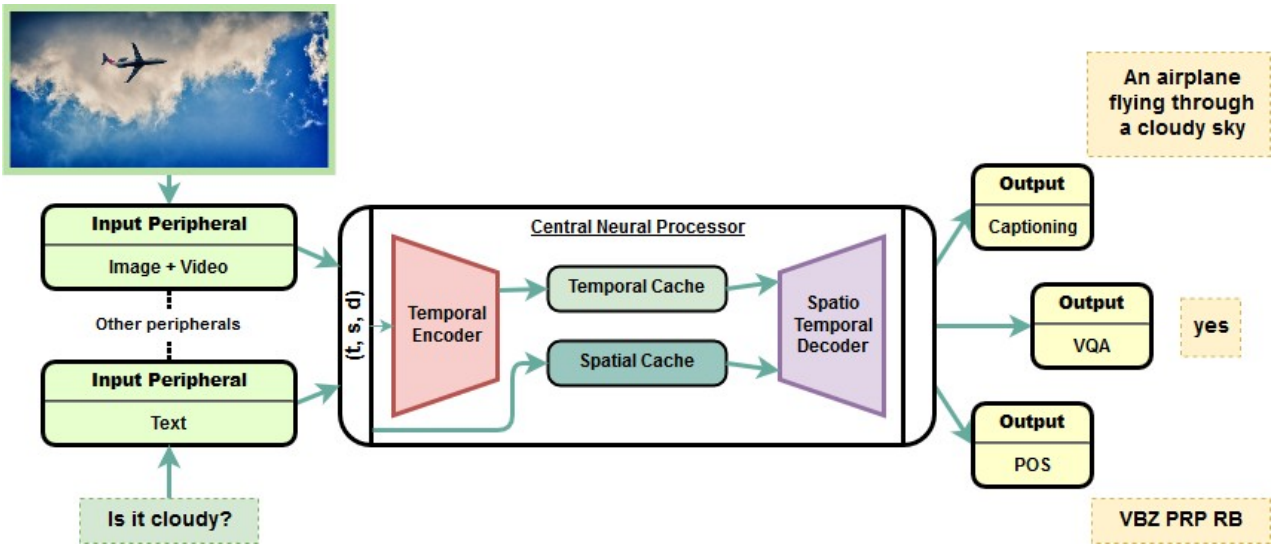


Şekil 3.15: Transformer mimarisi [1].

Kod çözücünün geleceğe göz atamamasını sağlamak kritik önem taşır. Bunu gerçekleştirmek için, anahtarlar ve sorgular kümesi her zaman adımında yalnızca geçmiş kelimeleri içerecek şekilde değiştirilebilir. Ancak bu çok verimsiz olacaktır. Bunun yerine, paralelleştirmeyi etkinleştirmek için, dikkat puanları $-\infty$ olarak ayarlanarak gelecekteki durumlar maskelenir. Kod çözücü işlemi de n kez tekrarlandıktan sonra, gömmeleri kelime dağarcığı uzunluğuna sahip daha büyük bir vektöre yansıtmak için doğrusal bir katman eklenir. Son olarak, bir softmax katmanı olası kelimeler üzerinde bir olasılık dağılımı oluşturur.

3.2 ÖNERİLEN MODEL

Birden fazla girdi alanı ile çok modlu görevlerin öğrenilmesini sağlamak ve herhangi bir görev kümesi için genel çoklu görevi desteklemek için birleşik bir mimari öneriyoruz. Mimarimiz, Merkezi Sinir İşlemcisi (CNP) adı verilen ortak bir merkezi sinir ağına bağlı çevresel ağlar olarak adlandırılan birden fazla alt ağdan oluşur (Şekil 3.16). Her bir çevresel ağ, alana özgü girdiyi özellik temsillerine kodlamak için kullanılır. Bu çalışmada görüntü, metin ve video çevre birimlerini tanımlıyoruz (Bölüm 3.2.1). Göreve bağlı olarak konuşma çevre birimi gibi daha fazlası da eklenebilir. Bir çevresel ağın çıktı temsili her zaman uzamsal-zamansal bir tensördür $x \in \mathbb{R}^{t*s*d_{model}}$; burada t ve s sırasıyla girdinin zamansal ve uzamsal boyutlarıdır ve d_{model} CNP'ye model boyutu girdisidir.



Şekil 3.16: Modelimiz görüntü altyazısı, görsel soru yanıtlama ve POS etiketlemeyi aynı anda gerçekleştiriyor

Her bir girdi alanına karşılık gelen çevresel ağlar tarafından üretilen uzamsal-zamansal temsiller daha sonra CNP tarafından işlenir. CNP, çoklu dil modelleme görevleri için en son teknoloji olan Transformer mimarisine [1] benzer şekilde dizi aktarımı için tamamen dikkat tabanlı kodlayıcı-kod çözücü [33, 42, 32] modelini kullanır (Bölüm 3.2.2). Kodlama aşamasında, CNP ilk olarak girdinin uzamsal-zamansal temsillerini işlemek ve saklamak için genel bir *encode(x, D)* işlevi uygular; burada $x \in \mathbb{R}^{t*s*d_{model}}$ çevresel ağlar tarafından üretilen uzamsal-

zamansal tensördür ve $D \in \mathbb{Z} : 0 \leq D < D_{len}$ etki alanı kimliğidir ve D_{len} CNP tarafından desteklenen maksimum etki alanı sayısıdır. *encode()* işlevi, ilgili çevre biriminden gelen her çok modlu girdi için bir kez olmak üzere birden çok kez çağrılır. Kod çözme aşamasında, tahminleri softmax olasılıkları olarak çözmek için bir $decode(y_{shifted}, \tau)$ işlevi kullanılır; burada $y_{shifted} \in \mathbb{Z}^{N-1}$ bir zaman adımı sağa kaydırılmış hedef çıktılardır, *N* çıktı dizisinin uzunluğudur; $\tau \in \mathbb{Z} : 0 \leq \tau < \tau_{len}$ görev kimliğidir ve τ_{len} desteklenen görevlerin toplam sayısıdır. Kod çözme adımı, uzamsal ve zamansal önbellek üzerinde iki aşamalı bir dikkat mekanizması içerecek şekilde değiştirilmiş [1]'e benzer.

3.2.1 **PERİFERİK AĞLAR (PERIPHERAL NETWORKS)**

Öncelikle, periferik ağları kullanarak birden çok giriş alanını nasıl desteklediğimizi ayrıntılı olarak anlatıyoruz. Bir periferik ağ, belirli bir alan girişini standart bir özellik temsiline $x \in \mathbb{R}^{t*s*d_{model}}$ kodlamak için var olan literatürden önceden eğitilmiş bir modeli kullanabilir, burada *t* ve *s* girişin sırasıyla zaman ve uzay boyutlarıdır ve d_{model}, Merkezi Sinir İşlemcisine giriş olarak verilen model boyutudur. Burada, metin ve görüş periferiklerini ayrıntılandırıyoruz ve göreve bağlı olarak daha fazla periferik ekleyebilir veya periferik tasarımını değiştirebilirsiniz. Bu sayede model birçok modaliteye daha ev sahipliği yapacak şekilde genişletilebilir.

Görüş periferiği: Bu periferik, görevlerdeki görüntü ve video girişlerini kodlamak için bir evrişimli sinir ağı kullanır. $h \times w \times n_c$ Boyutunda bir resim için, bu periferik $h' \times w' \times n_c'$ boyutlarına örnek alır, burada h, w, n_c sırasıyla yükseklik, genişlik ve giriş kanallarının sayısıdır. Bir video için, her kare periferiğe giriş olarak verilir ve $F \times h' \times w' \times n_c'$ üretir, burada F videodaki toplam kare sayısıdır. Kodlama vektörleri daha sonra tamamen bağlı bir katman kullanılarak d_{model} boyutuna yansıtılır. Çıktı daha sonra bir uzay-zaman tensörü olan $x \in \mathbb{R}^{t*h'*\omega'*d_{model}}$, şeklinde yeniden şekillendirilir, burada t =1 bir resim için ve t =F bir video için. Deneylerimizde, 152 evrişimli katmana sahip bir ResNet [43] varyantı olan önceden eğitilmiş ResNet-152 modelini kullanıyoruz. Bir resim/video

için mekansal özellik temsillerini oluşturmak için son fully connected ve avg-pooling katmanlarını kaldırıyoruz.

Dil periferiği: Dil periferiği, verilen bir giriş cümlesi için alt kelimeler oluşturmak için bayt çifti kodlamasını [44] kullanır. Alt kelimeler, d_{emb} boyutunda alt kelime gömülülerini oluşturmak için bir gömme katmanına geçirilir ve bir fully connected katman kullanılarak dmodel boyutuna yansıtılır. Çıktı daha sonra bir uzamsal-zamansal tensörü $x \in \mathbb{R}^{t*1*d_{model}}$, şeklinde yeniden şekillendirilir, burada *t* giriş cümlesindeki alt kelimelerin sayısına eşittir. Metinsel verilerde herhangi bir uzaysal boyutumuz olmadığı için, bir Dil periferiğinden gelen *x*'in uzaysal boyutu her zaman 1'dir. Deneylerimizde, d_{emb} = 300 ve *vocab_size* = 25000 olan önceden eğitilmiş alt kelime gömülerini kullanıyoruz [45], bu, gömme matrisinin ağırlıklarını başlatmak için 275'ten fazla dilin önceden eğitilmiş alt kelime gömülülerini içerir.

3.2.2 **MERKEZİ SİNİR İŞLEMCİSİ (CNP)**

Giriş verilerindeki uzamsal-zamansal bilgileri işlemek için CNP bir uzamsal önbellek C_s, zamansal önbellek C_t ve bir bağlantı dizisi *L* uygular. Uzamsal ve zamansal önbellek ve bağlantı dizisi, kodlama işleminden önce boş olarak başlatılan bir öğe listesidir. Kodlama aşamasında, bir *encode()* rutini girdi olarak çevre biriminden üretilen tensör *x*'i ve ilgili etki alanı/çevre birimi kimliği *D*'yi alır. Bu işlev, girdi *x*'teki uzamsal ve zamansal bilgileri işler ve bunları sırasıyla uzamsal önbellek C_s ve zamansal önbellek C_t'ye depolar ve boyutlarını *t* & *s* bağlantı dizisinde saklar. Belirli bir görev için bu *encode()* rutini *K* kez çağrılır, burada *K* görevdeki girdi sayısıdır. Bu girdilerin aynı veya farklı alanlara ait olabileceğini unutmayın.

Şekil 3.17: ***soldaki****: Modelimizin TemporalEncoder mimarisi;* ***sağdaki****: Modelimizin decode() mimarisi.*

Encode(x, D): Verilen bir $x \in \mathbb{R}^{t*s*d_{model}}$ girdisi ve D etki alanı tanımlayıcısı için *encode()* rutini Algoritma 1'de açıklanmıştır. Girdiler birden fazla çevre biriminden gelebileceğinden, algoritma ilk olarak girdinin etki alanına duyarlı bir şekilde kodlanmasını sağlamak için girdiyi etki alanı gömme ile birleştirir (Adım 2 ile 3). Adım 4 ila 7, zaman boyutunu açarak ve bu açılmış vektörleri uzamsal önbelleğe ekleyerek *x*'teki uzamsal bilgileri işler. Adım 8 ila 10,

x'in uzamsal boyutunun ortalamasını alarak ve ardından ortalaması alınmış tensörü öz dikkat tabanlı bir TemporalEncoder'a aktararak *x*'teki zamansal bilgiyi işler. Bu TemporalEncoder, Şekil 3.17'de gösterildiği gibi [1]'de kullanılan kodlayıcıya benzer ve giriş dizisinin zamansal gömülmelerini hesaplamak için kullanılır. TemporalEncoder'dan elde edilen çıktı temporal önbelleğe eklenir.

Algoritma 1 *encode()*: Uzamsal ve zamansal temsilleri uzamsal ve zamansal önbelleğe kodlar

Require: $x \in \mathbb{R}^{t*s*d_{model}}$, D, C_S, L, Ct
1: $L \leftarrow L \cup (t \rightarrow s)$
2: $D_{emb} \leftarrow EmbedLayer(D)$
3: $x \leftarrow FC(Concat(x, D_{emb}), d_{model})$
4: **if** $s > 1$ **then**
5: $S \leftarrow Reshape(x, (ts, d_{model}))$ {burada, çıktı $S = [S_1, \ldots, S_{ts}]$ s.t. $S_i \in \mathbb{R}^{d_{model}}$ Bir uzamsal özellik vektörüdür}
6: $C_S \leftarrow C_S \cup [S1, \ldots, Sts]$ {Uzamsal temsilleri uzamsal önbelleğe eklenir}
7: **end if**
8: $T \leftarrow (\sum_{i=1}^{s} x[:, i, :]) / s$
9: $T \leftarrow TemporalEncoder(T)$ {Burada, çıktı $T = [T_1, \ldots, T_t]$ s.t. $T_j \in \mathbb{R}^{d_{model}}$ x'teki zamansal boyutun kodlanmasıdır}
10: $C_t \leftarrow C_t \cup [T_1, \ldots, T_t]$ {Zamansal temsilleri zamansal önbelleğe eklenir}

Yukarıdaki kodlama rutini, her $x_k \in \mathbb{R}^{t_k*s_k*d_{model}}$ girişi için Ct & C_S 'ye uzamsal-zamansal bilgi eklemeye devam eder. Görevin *k*'ncı girdisine karşılık gelen üst simge *k*'ya dikkat edin; burada $k \in 1, \ldots, K$. K defa çağrıdan sonra, uzamsal önbellek $C_t = [T_1, \ldots, T_R]$, burada $R = \sum_{r=1}^{K} t_r$ zamansal önbellek $C_s = [S_1, \ldots, S_p]$, burada $P = \left\{ \sum_p t_p * s_p : \ p \in 1, \ldots, K \wedge s_p > 1 \right\}$ ve bağlantı dizisi $L = [(t_1 \rightarrow s_1), \ldots, (t_k \rightarrow s_k)]$. C_s'nin *encode()* işlevinin $s_k = 1 \forall k$ olan girdilerle çağrılması durumunda boş olabileceğini unutmayın. Daha sonra da tahminleri softmax olasılıkları olarak oluşturmak için *decode()* rutinini kullanacağız.

Decode($y_{shifted}$, τ): *decode()* fonksiyonunun mimarisi Şekil 3.17'de gösterilmektedir. *Decode()*, bir zaman adımı sağa kaydırılmış $y_{shifted}$ çıktı etiketlerini, bir görev kimliği τ'yu argüman olarak alır ve uzamsal ve zamansal önbellekten katılarak tahminler üretir. *Decode()* işlevi, Transformer mimarisinde [1,2] kullanılan kod çözücüye benzer şekilde yapılandırılmıştır ve zamansal ve uzamsal önbellekte depolanan vektörlere birlikte katılır. [1]'e benzer şekilde, kod

çözme ilk olarak maskelenmiş çok kafalı ölçeklendirilmiş nokta çarpımı dikkatini kullanarak çıktı gömülmelerine dikkat ederek başlar. Zamansal önbellek için dikkat katmanı, [1]'de belirtildiği gibi çok başlı ölçeklendirilmiş nokta çarpımı dikkatini kullanır. Uzamsal önbellek için dikkat katmanı, uzamsal önbelleğin öğelerine katılmak için geçitli çok kafalı dikkat (gated multi-head attention) kullanır. Hem zaman hem de uzay boyutuna sahip girdiler için (örneğin video), uzamsal dikkat katmanının zamansal önbellek dikkat katmanında nispeten yüksek dikkat puanlarına sahip karelere daha fazla katılmasını istiyoruz. Bu nedenle, zamansal önbellek çok kafalı dikkat katmanından elde edilen dikkat puanı çıktısı $A \in \mathbb{R}^{n_h * N * R}$, uzamsal dikkat katmanında dikkat puanı çıktısını geçitlemek için kullanılan $G \in \mathbb{R}^{n_h * N * P}$ tensörünü hesaplamak için kullanılır; burada n_h, [1]'de açıklandığı gibi çok kafalı dikkatteki kafa sayısıdır. G tensörü, Algoritma 2'de ayrıntılı olarak açıklandığı gibi A & L kullanılarak hesaplanır. [1]'deki notasyonun aynısını kullanarak Q: sorgu matrisi, K: d_k boyutun anahtarları ve V: d_v boyutun değerleri, uzamsal katman için ölçeklendirilmiş nokta çarpımı dikkati şu şekilde değiştirdik:

$$Attention(Q, K, V, G) = \left(softmax\left(\frac{Qk^T}{\sqrt{dv}}\right) \odot G \right) V \qquad (1)$$

Algorithm 2 Zamansal dikkat ve bağlantı dizisinden elde edilen çıktı puanlarını kullanarak G'yi hesaplayın

Require: L, A

$idx \leftarrow 0$

$G \leftarrow []$
if $s > 0$ **then**
  $A' \leftarrow A[:, :, idx : idx + t]$
  $A' \leftarrow Expand(A', (n_h, N, t, s))$ {burada $Expand(tensor, dimension)$ tensörü belirli bir boyuta göre genişletir}
  $A' \leftarrow Reshape(A', (n_h, N, ts))$
  $G \leftarrow G \cup A'$ {İlgili zamansal dikkat puanlarını G'ye eklenir}
end if
$idx \leftarrow idx + t$
end for
$G \leftarrow Stack(G)$ {Boyutlu G tensörünü oluşturmak için tensör listesini yığınlanır (n_h, N, P)}

Aynı CNP'yi farklı çıktı sözcüklerine sahip birden fazla görevle kullanmak

amacıyla ve her görev için çıktı gömülerini oluşturmak için, birden fazla çıktı gömme katmanı $OutputEmbedLayer_1, \ldots, OutputEmbedLayer_{\tau_{len}}$ kullanıyoruz. Son katmanda, birden fazla (FullyConnected + Softmax)1, ..., (FullyConnected + Softmax)τlen sınıflandırma katmanlarını her görev için kullanıyoruz. Ayrıca τ kullanarak bir görev gömme vektörü hesaplıyoruz ve her zaman görev gömme vektörünü kullanarak kod çözmeye başlıyoruz.

3.2.3 **ÇOK GÖREVLİ ÖĞRENME (MTL)**

Tek bir modeli aynı anda birden fazla görev üzerinde eğitmek için [46]'da açıklandığı gibi HogWild eğitim yaklaşımını kullandık. [47]'de açıklanan yaklaşıma benzer şekilde, ana süreç modelin global bir kopyasını tutar. Her görev için ayrı işçi süreçleri oluşturuyoruz ve her süreç modelin yerel bir kopyasını tutuyor. Her eğitim iterasyonunda, her süreç kendi yerel modelini global kopya ile senkronize ederek başlar. Bu, yerel kopyasında ileri ve geri yayılma yoluyla ve ardından yerel olarak hesaplanan gradyanları eş zamansız olarak küresel modele kopyalayarak yapılır. Her işlem daha sonra global modelin ağırlıklarını güncellemek için global model optimize ediciyi eş zamansız olarak çağırır. Modeli [47]'de olduğu gibi CPU'da depolamak yerine yerel kopyaları her zaman GPU'da depoluyoruz, birden fazla GPU kullanılması bu işlemin tam gerekliliğini yerine getirecektir.

4. GÖREVLER VE GERÇEKLEŞTİRİM

Farklı modaliteleri kapsayan görevler için önerilen çerçevemizin etkinliğini değerlendirmek amacıyla, tüm olası uzamsal-zamansal veri arketiplerini kapsayan bir set seçtik: Resim Altyazısı, Konuşma Parçası (POS) etiketleme, Görsel Soru Yanıtlama (VQA) ve Video-aktivite Tanıma. Bu görevlerin her biri, *t* ve *s*'nin sırasıyla girdinin zamansal ve uzamsal boyutları olduğu farklı *t* ve *s* değerlerini içeren girdinin benzersiz bir potansiyel uzamsal-zamansal yapılandırmasını araştırır. Bu, sistemin çok modlu ve çoklu girdi yetenekleri hakkında kapsamlı bir çalışma yapmamızı sağlar. Bu görevlerin özelliklerini aşağıda daha ayrıntılı olarak açıklıyoruz.

Eğitim için her zaman Adam optimizer [48] ile çapraz entropi kaybı kullanıyoruz ve öğrenme oranını [1]'e benzer şekilde Noam scheduler [22] kullanarak planlıyoruz. Görsel çevre biriminde, önceden eğitilmiş ResNet modelinin katmanlarını donduruyoruz. Mimarinin geri kalan kısımları (çevre birimleri ve CNP) eğitilebilir durumda tutulmaktadır. Bu bölümde, kullanılan veri kümeleri ve bu görevlerin her biri için model kurulumu hakkında ayrıntılar sunuyoruz.

Konuşma Parçası (POS) Etiketleme: Görevi yalnızca zamansal modalite ile göstermek için *(t > 1 & s = 1*, burada *t* ve *s* sırasıyla girdinin zamansal ve uzamsal boyutlarıdır), POS etiketleme problemini ele alıyoruz. Bir girdi kelime dizisi verildiğinde, model her bir kelimeye karşılık gelen bir dizi POS etiketi üretmelidir. Uzmanlar tarafından İngilizce WSJ makaleleri üzerine altın ek açıklamalar içeren Penn Tree-bank [16] kullanıyoruz. Kodlama aşamasında, her bir girdi cümlesi dil çevre birimi tarafından işlenerek bir uzamsal-zamansal tensör $x \in \mathbb{R}^{t*1*d_{model}}$ oluşturulur; burada *t*, girdideki alt kelimelerin dizi uzunluğudur. CNP *encode()* işlevi daha sonra *x*'i zamansal önbelleğe kodlamak için kullanılır. Uzamsal önbelleğin metin girdileri için s = 1 olarak boş olduğunu unutmayın. Bu nedenle, kod çözme aşaması POS etiketlerinin sırasını tahmin etmek için Transformers ile aynıdır.

Resim Altyazısı Oluşturma: Bu görev, yalnızca uzamsal modalite *(t = 1 & s > 1)* içeren girdilere sahip olanları temsil eder. Resim yazısı modelinin belirli bir

görüntü için metin başlığını tahmin etmesi gerekmektedir. Eğitim için MSCOCO 2014 veri setini [49] kullandık ve sonuçları COCO doğrulama seti üzerinde sunduk. Kodlama aşamasında, giriş görüntüsü 224 224 olarak yeniden boyutlandırılır ve görüntü gömme $x \in \mathbb{R}^{1*49*d_{model}}$ üretmek için önceden eğitilmiş ResNet-152 içeren görsel çevre birimi tarafından işlenir. *x* daha sonra ilgili uzamsal ve zamansal önbelleği dolduran *encode()* işlevine girdidir. Kod çözme aşaması, altyazıları oluşturmak için çıktı kelime boyutu 25000 olan *decode()* işlevini kullanır.

Görsel Soru Yanıtlama: Her biri uzamsal veya zamansal modalite içeren (her girdi için *t > 1 & s = 1* veya *t = 1 & s > 1*) çoklu alan (multiple domain) girdileri olan görev için görsel soru cevaplama görevini seçiyoruz. Girdi olarak bir görüntü üzerinden bir soru verildiğinde, modelin doğru cevap etiketini tahmin etmesi beklenir. Bu amaç için VQA v2.0 veri setini [50] kullandık ve değerlendirmeyi VQA test-dev seti üzerinde gerçekleştirdik. Tüm görüntüler eğitimden önce 224 224 boyutuna yeniden boyutlandırılmıştır. Bu görevin kodlama aşamasında iki çevre birimi kullanılır: görsel çevre birimi, giriş görüntüsü için bir tensör $x_1 \in \mathbb{R}^{1*49*d_{model}}$ oluşturmak için kullanılır. Dil çevre birimi, soruları $x_2 \in \mathbb{R}^{t*1*d_{model}}$ 'e kodlamak için kullanılır; burada *t*, sorudaki alt kelimelerin uzunluğuna eşittir. *encode()* işlevi, ilk olarak x_1 ve ikinci olarak x_2 girdi olarak olmak üzere iki kez çağrılır. Son olarak, 3500 kelime boyutuna sahip *decode(),* cevapları tek bir kod çözme adımında *softmax* olasılıkları olarak üretmektir.

Video Etkinliği Tanıma: Tek bir girdide hem uzamsal hem de zamansal modalite içeren görevler için *(t > 1* ve *s > 1)*, videolar üzerinde eylem tanıma görevini ele alıyoruz. Bu amaçla HMDB veri kümesini kullanıyoruz [51]. Veri kümesi, 51 sınıflı 5000'den fazla gerçek yaşam eyleminin kısa uzunluktaki kliplerinden oluşmaktadır. Video başına 16 kare kullanıyoruz ve her birini 224 224 olarak yeniden boyutlandırıyoruz. Kodlama aşamasında, videonun her bir karesi, daha sonra *encode()* işlevine girdi olarak kullanılan bir video kodlama $x \in \mathbb{R}^{16*49*d_{model}}$ 'ini kümülatif olarak oluşturmak için görüş çevre biriminden geçirilir. Son olarak, çıkış kelime haznesi boyutu 51 olan *decode()*, eylemi tek bir kod çözme adımında *softmax* olasılıkları olarak tahmin etmektir.

5. BULGULAR VE DENEYLER

Değerlendirmeyi (a) Bölüm 4'te gösterilen çeşitli modalitelerin görevleri (b) Bu görevler için çoklu görev kurulumu (Tablo 5.1) (c) Çoklu görev modelinin görünmeyen bir görev için yeniden kullanımı (Şekil 5.1) üzerinde sunuyoruz. Ayrıca, mimari üzerinde yapılan bazı ablasyon (Bir ablasyon çalışması, bileşenin genel sisteme katkısını anlamak için belirli bileşenleri kaldırarak bir YZ sisteminin performansını araştırır.) çalışmalarını da sunuyoruz (Tablo 5.2).

Önerilen mimarinin bireysel görevler üzerindeki performansı: Önceki Bölüm 4'te açıklandığı gibi çeşitli girdi modaliteleri ve kombinasyonlarına sahip dört görevden oluşan bir set seçiyoruz. Modelimizi yukarıdaki görevlerin her birinde bağımsız olarak eğitiyoruz. Görevlerin her biri bu genel mimarinin benzersiz yeteneklerini göstermektedir. Daha spesifik olarak, Tablo 5.1'de sonuçlarımızı aynı datasette eğitilmiş en başarılı aşağıdaki son teknolojiler ile karşılaştırıyoruz: - POS etiketleme: [53]; resim altyazısı ve VQA: [52] ve HMDB: [54]. Herhangi bir hiper parametre optimizasyonu yapmadığımızı belirtmek önemlidir. Daha fazla hesaplama gücüyle, hiper parametrelerin bu görevlere yönelik ince ayarının, en son teknolojiyle karşılaştırılabilir veya hatta iyileştirilmiş performansla sonuçlanacağına inanıyoruz. Bu sonuçlar, çeşitli olası uzamsal-zamansal arketiplerde tek bir mimari kullanmayı amaçlayan gelecekteki çalışmalar için bir temel olarak kullanılabilir. Modelin, alan girdilerini uzamsal-zamansal tensörlere dönüştürmek için belirli bir çevre birimi eklenebildiği sürece CNP'de herhangi bir değişiklik yapılmadan yeni bir alana genişletilebilir olduğunu belirtmek gereklidir. Mimarinin bu yönü, onu birçok popüler çok modlu göreve uygulanabilir kılmaktadır.

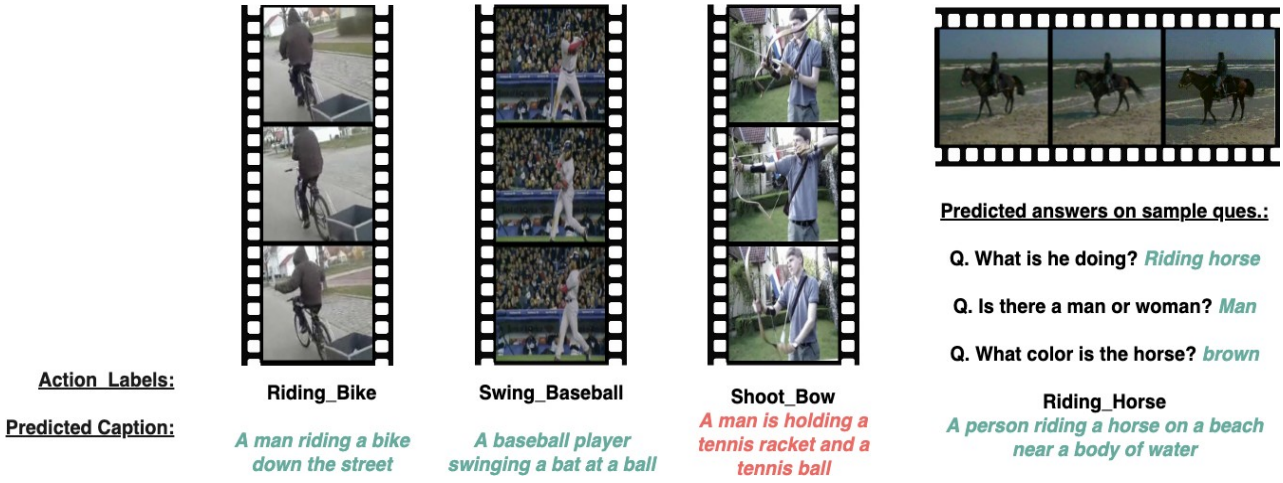
Tablo 5-1: Modelimizin çeşitli görevler üzerindeki performansı. IND: Verilen görevlerin her biri için ayrı ayrı eğitilen model; MULT-3: POS, Altyazı ve VQA üzerinde eğitilen çoklu görev modeli; MULT-4: Dört görevin tamamında eğitilen çoklu görev modeli. SOTA: state of the art

	POS	**Captioning**		**Visual Question Answering**				**HMDB**	**#PARAMS**
	Acc.	BLEU-4	Meteor	Overall	Y/N	Num.	Other	Acc.	
SOTA	97.44	36.2	27.0	63.2	80.3	42.8	55.8	59.4	-
IND	94.92	29.1	24.8	55.27	74.05	35.17	46.55	55.29	$450m$
MULT-3	96.03	28.9	24.8	56.82	76.54	36.11	47.23	-	$149.05m$
MULT-4	95.56	27.6	24.7	55.89	75.33	35.59	45.94	54.32	$149.17m$

Farklı görevlerin bir arada eğitilmesinin etkisi: İki çoklu görev modelini eğittik: (1) MULT-3 (POS+VQA+Captioning) ve (2) MULT-4 (POS+VQA+Captioning+HMDB). MULT-3 modeli benzer ve bazen daha iyi performans elde ederken, nihai MULT-4 modeli bağımlı görev puanlarıyla karşılaştırıldığında biraz daha düşük performans elde etmektedir. Bunun, yalnızca 5000 eğitim örneği içeren HMDB veri kümesinin boyutundaki çarpıklıktan kaynaklandığına inanıyoruz. Bununla birlikte, bir değiş tokuş olarak, HMDB görevinin eklenmesi, düşük gösterilen ilginç sıfır-atış (zero-shot) sonuçları göstermektedir. Çoklu görev modelinin kullanılması da toplam parametre sayısının üç kat azalmasını sağlamaktadır. Yani, her görev için ayrı bir model kullanıldığında, toplam 450×10^6 parametreye sahip oluyoruz. Oysa çoklu görev sırasında tek bir model paylaşıldığından, benzer performans elde ederken toplamda 149×10^6 parametreye sahip oluyoruz. İlginç bir şekilde, model farklı görevlerden gelen girdilerin uzamsal-zamansal bileşenlerine katılabiliyor ve bunlar arasında uyumlu tahminler üretebiliyor, böylece mimarimizin genelleme yeteneğini gösteriyor.

Sıfır atışlı öğrenmeye doğru: önceden eğitilmiş ağın görülmeyen görevler için yeniden kullanımı: Temsillerin birden fazla görev arasında paylaşılması, faydalı bilginin birden fazla alana aktarılmasına fayda sağlar. Görüntü ve video aynı görme çevresi tarafından işlendiğinden, dört görevin tümü (MULT-4) üzerinde önceden eğitilmiş modelimizin, bu görevler üzerinde herhangi bir açık eğitim olmaksızın video altyazısı ve video soru cevaplama gerçekleştirip gerçekleştiremeyeceğini, yani sıfır atışlı öğrenmeyi görmek için bir deney yaptık. HMDB test bölümü 1'den rastgele seçilen örnekler üzerinde yapılan değerlendirmenin sonuçları Şekil 5.1'de gösterilmektedir. İlginç bir şekilde, model COCO ve VQA eğitim setinde bulunan ilgili eylemlerde oldukça iyi performans göstermektedir; örneğin binicilik ve beyzbolla ilgili altyazılar veya VQA'da bulunan kavramlarla ilgili sorular. Model, herhangi bir video altyazısı ve video QA örneği üzerinde eğitim almadan, görüntü altyazısı, görüntü QA (VQA) ve video eylem tanıma (HMDB) eğitimli bilgilerini anlamlı tahminler oluşturmak için videolara uygulayabilir, böylece modelin ilgili çok modlu görevler arasında bilgi aktarma yeteneğini gösterir. Bununla birlikte, eğitilmiş veri kümelerinde bulunmayan kavramlar üzerinde, model ya videodaki ortamı tanımlar ya da

alternatif bilinen kavramlarla değiştirir. Bu vaka çalışması, kapsamlı olmasa da modelin paylaşılan temsilleri öğrenme ve bilgiyi alanlar arasında aktarma kabiliyetini göstermektedir. Daha fazla görev ve etki alanı eklenmesinin gelecekte çok çeşitli problemlerde daha ilginç zero-shot öğrenme sonuçlarına yol açacağına inanıyoruz.



Şekil 5.1: Sıfır çekim video altyazı ve video soru-cevaplama sonuçları.

Bireysel mimari bileşenlerin etkisi: Farklı girdi alanlarını desteklemek için mimarimiz, orijinal Transformer mimarisine (yalnızca zamansal verileri işleyen mekanizmalardan oluşan) uzamsal önbellek ve bağlantı dizisi bileşenlerini ekler. Tablo 5.2'de gösterildiği gibi çeşitli görevlerdeki önemlerini doğrulamak için bu bileşenlerin her biri üzerinde bir ablasyon çalışması gerçekleştirdik. Ablasyon bağımsız (IND) ve çoklu görev modeli (MULT-4) üzerinde gerçekleştirilmiştir. İkinci satırda mimarimizden bağlantı dizisi çıkarılmıştır, yani Denklem 1'deki G çarpımı kaldırılmıştır. Bağlantı dizisi, tek bir girdide hem uzamsal hem de zamansal modalite içeren video gibi girdilere sahip görevlere yardımcı olmak için tasarlanmıştır. Videodaki kare sayısı arttıkça toplam uzamsal bileşen sayısı çok artar ve bu da video boyunca çeşitli uzamsal bölgelere dikkat etmeyi zorlaştırır. Bağlantı dizisi kullanılarak uzamsal dikkat katmanı videodaki belirli önemli karelere daha fazla katılabilir. Bu nedenle, bağlantı dizisinin kaldırılması, herhangi bir girdi için her iki uzamsal-zamansal modaliteye sahip olmadıklarından, HMDB'de diğer görevlere kıyasla performansta büyük bir düşüşe neden olur. Öte yandan, uzamsal önbelleğin kaldırılması, uzamsal modalite içeren tüm görevlerde performans üzerinde önemli bir etkiye sahiptir. Resim altyazıları öncelikle uzamsal

modalite içerdiğinden, önbelleğin kaldırılmasından sonra BLEU önemli ölçüde düşmektedir. Diğer görevler tahmin için zamansal önbelleği kullandığından, çoklu görev ortamında altyazı görevi tahmin için zamansal önbellekte depolanan görüntünün uzamsal ortalamasını kullanmayı öğrenir ve bu nedenle uzamsal önbelleğin kaldırılmasından sonra bir miktar performans korur. Öte yandan, altyazı yazma konusunda bağımsız olarak eğitildiğinde, ağ yalnızca uzamsal önbellekteki bilgileri kullanmayı öğrenir ve bu nedenle puan sıfıra düşer. VQA hem görüntüden gelen uzamsal bilgiyi hem de sorudan gelen zamansal bilgiyi kullanır, zamansal önbelleğin kullanımından bir miktar performans elde eder. POS etiketleme görevinin, girdide yalnızca zamansal modaliteye sahip olduğu için bileşenlerden herhangi birinin çıkarılmasından etkilenmediğini unutmayın.

	POS (Acc.)		**Captioning** (BLEU-4)		**VQA** (Overall)		**HMDB** (Acc.)	
	IND	MULT-4	IND	MULT-4	IND	MULT-4	IND	MULT-4
Default	94.92	95.56	29.1	27.6	55.27	55.89	55.29	54.32
w/o link array	95.61	95.44	28.9	27.4	54.05	55.30	45.94	46.79
w/o spatial cache	95.61	95.44	0	11.9	39.86	44.24	10.91	11.50

Tablo 5-2: Önerilen mimari bileşenlerin etkisi üzerine ablasyon çalışması.

6. SONUÇ VE İLERİ ÇALIŞMALAR

Sistem Farklı modalitelerde çoklu girdilere sahip görevleri öğrenebilen birleşik bir sinir ağı mimarisi sunuyoruz. Mimari, uzamsal-zamansal veriler içeren herhangi bir görev kümesinde çoklu görev öğrenimi için daha fazla benimsenebilir. Bir modelin birden fazla görevde paylaşılması, toplam parametre sayısında da önemli bir azalma sağlar. Ayrıca, bu paylaşılan modelin, görülmeyen görevler için yeniden kullanılabilen çeşitli uzamsal-zamansal girdilerden sağlam temsiller öğrenebileceğini gösteriyoruz. Önerilen bu mimarinin, uzamsal-zamansal girdileri olan herhangi bir görev için geniş bir uygulanabilirliğe sahip olduğuna inanıyoruz. Kullanılabilirliğini artırmak için, konuşma gibi daha fazla alanı destekleyen yeni çevre birimleri sunmak istiyoruz. Ayrıca grafikler ve ilişkisel veriler gibi zamansal ve uzamsal boyutların ötesinde verilerin diğer yönlerini keşfetmeye de hevesliyiz. Ayrıca, çoklu görev çerçevesini optimize etmek için zamanlama mekanizmalarını araştırmak ilginç olacaktır.

KAYNAKÇA

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 5998–6008. Curran Associates, Inc. (2017).

[2] R. Hu, A. Singh. UniT: Multimodal Multitask Learning with a Unified Transformer. arXiv preprint arXiv:2102.10772, 2021.

[3] Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR abs/1602.07261 (2016).

[4] Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., ukasz Kaiser, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., Dean, J.: Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144 (2016).

[5] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., Ng, A.Y.: Multimodal deep learning. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. pp. 689–696. ICML'11, Omnipress, USA (2011).

[6] Collobert, R., Weston, J.: A unified architecture for natural language process- ing: Deep neural networks with multitask learning. In: Proceedings of the 25th International Conference on Machine Learning. pp. 160–167. ICML '08, ACM, New York, NY, USA (2008).

[7] Hao Tan and Mohit Bansal. Vokenization: improving language understanding with contextualized, visual-grounded supervision. arXiv preprint arXiv:2010.06775, 2020.

[8] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In Proceedings of CVPR, pages 10437–10446, 2020.

[9] Hao Hao Tan and Mohit Bansal. Lxmert: Learning crossmodality encoder representations from transformers. In Proceedings of EMNLP/IJCNLP, 2019.

[10] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pre-training for image captioning and vqa. In AAAI, pages13041–13049, 2020.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[12] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. Endto-end object detection with transformers. In Proceedings of ECCV, 2020

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT, 2019.

[14] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. arXiv preprint arXiv:1908.03557, 2019.

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683, 2019.

[16] Marcus, M., Kim, G., Marcinkiewicz, M.A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The penn treebank: Annotating predicate argu- ment structure. In: Proceedings of the Workshop on Human Language Technol- ogy. pp. 114–119. HLT ’94, Association for Computational Linguistics, Strouds- burg, PA, USA (1994).

[17] Battenberg, E., Chen, J., Child, R., Coates, A., Gaur, Y., Li, Y., Liu, H., Satheesh, S., Seetapun, D., Sriram, A., Zhu, Z.: Exploring neural transducers for end-to- end speech recognition. CoRR abs/1707.07413 (2017).

[18] Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications. OpenAI Blog https://openai.com/blog/better-language-models , 2019.

[19] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R

Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Proceedings of NeurIPS, pages 5753–5763, 2019.

[20] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019.

[21] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942, 2019.

[22] Shazeer, N., Stern, M.: Adafactor: Adaptive learning rates with sublinear memorycost. CoRR abs/1804.04235 (2018).

[23] C Rosset. Turing-NLG: A 17-billion-parameter language model by microsoft. Microsoft Blog, 2020.

[24] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. arXiv preprint arXiv:1802.05751, 2018.

[25] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In International Conference on Machine Learning, pages 1691–1703. PMLR, 2020.

[26] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. arXiv preprint arXiv:1706.05137, 2017.

[27] Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. Learning visually grounded sentence representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 408–418, 2018.

[28] Devendra Singh Chaplot, Lisa Lee, Ruslan Salakhutdinov, Devi Parikh, and Dhruv Batra. Embodied multimodal multitask learning. arXiv preprint arXiv:1902.01385, 2019.

[29] Subhojeet Pramanik, Priyanka Agrawal, and Aman Hussain. Omninet: A unified architecture for multi-modal multi-task learning. arXiv preprint arXiv:1907.07804, 2019.

[30] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. arXiv preprint arXiv:1911.03584, 2019.

[31] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In Proceedings of ACL, 2019.

[32] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks, (2014).

[33] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, (2014).

[34] Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, (2019). *CoRR*, abs/1905.11946.

[35] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations, (2020).

[36] Pilehvar, M. T. and Camacho-Collados, J. (2021). Embeddings in Natural Language Processing. Springer International Publishing.

[37] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space, (2013).

[38] Google (2022). Embeddings: Translating to a lower-dimensional space. https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space .

[39] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information, (2016).

[40] Manning, C., Goldie, A., and Hewitt, J. (2022). Stanford cs224n: Natural language processing with deep learning. https://web.stanford.edu/class/cs224n/slides/ .

[41] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B.,Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. (2021).

Highly accurate protein structure prediction with AlphaFold. 596(7873):583–589.

[42] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (2014).

[43] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016).

[44] Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (2016).

[45] Heinzerling, B., Strube, M.: BPEmb: Tokenization-free Pre-trained Subword Em- beddings in 275 Languages. In: chair), N.C.C., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Hasida, K., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S., Tokunaga, T. (eds.) Proceedings of the Eleventh Interna- tional Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (2018)

[46] Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: A lock-free ap- proach to parallelizing stochastic gradient descent. In: Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 24, pp. 693– 701. Curran Associates, Inc. (2011).

[47] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Harley, T., Lillicrap, T.P., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. pp. 1928–1937. ICML'16, JMLR.org (2016).

[48] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015).

[49] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Doll´ar, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing, Cham (2014)

[50] Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D.: Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2017)

[51] Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: Proceedings of the International Conference on Computer Vision (ICCV) (2011)

[52] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L.: Bottom-up and top-down attention for image captioning and visual question answering. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6077–6086 (2018).

[53] Spoustov´a, D.j., Haji˘c, J., Raab, J., Spousta, M.: Semi-supervised training for the averaged perceptron POS tagger. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009). pp. 763–771. Association for Computational Linguistics, Athens, Greece (2009).

[54] Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 568–576. Curran Associates, Inc. (2014).