

# Evrişimli Sinir Ağları El Kitabı VIP

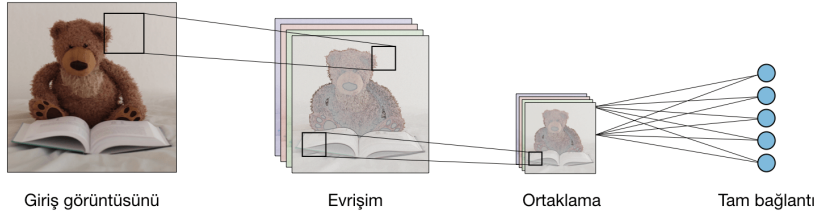
Afshine AMIDI ve Shervine AMIDI

April 30, 2019

Ayyüce Kızrak ve Yavuz Kömeçoğlu tarafından çevrilmiştir

## Genel bakış

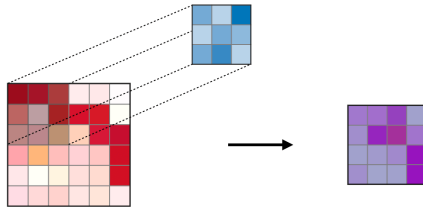
□ **Geleneksel bir CNN (Evrişimli Sinir Ağı) mimarisi** – CNN'ler olarak da bilinen evrişimli sinir ağları, genellikle aşağıdaki katmanlardan oluşan belirli bir tür sinir ağıdır:



Evrişim katmanı ve ortaklama katmanı, sonraki bölümlerde açıklanan hiperparametreler ile ince ayar (fine-tuned) yapılabilir.

## Katman tipleri

□ **Evrişim katmanı (CONV)** – Evrişim katmanı (CONV) evrişim işlemlerini gerçekleştiren filtreleri,  $I$  girişini boyutlarına göre tararken kullanır. Hiperparametreleri  $F$  filtre boyutunu ve  $S$  adımı içerir. Elde edilen çıktı  $O$ , öznetelik haritası veya aktivasyon haritası olarak adlandırılır.

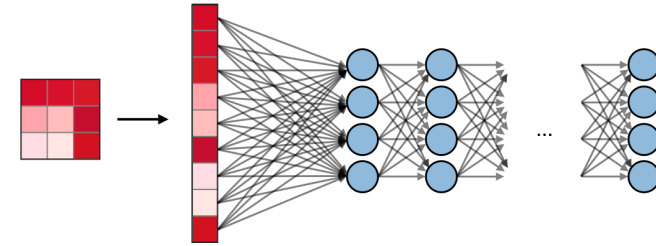


Not: evrişim adımı, 1B ve 3B durumlarda da geliştirilebilir ( $B$ : boyut).

□ **Ortaklama (POOL)** – Ortaklama katmanı (POOL), tipik olarak bir miktar uzamsal değişkenlik gösteren bir evrişim katmanından sonra uygulanan bir örnekleme işlemidir. Özellikle, maksimum ve ortalama ortaklama, sırasıyla maksimum ve ortalama değerin alındığı özel ortaklama türleridir.

	Maksimum ortaklama	Ortalama ortaklama
<b>Amaç</b>	Her ortaklama işlemi, geçerli matrisin maksimum değerini seçer	Her ortaklama işlemi, geçerli matrisin değerlerinin ortalaması alır
<b>Görsel Açıklama</b>		
<b>Açıklama</b>	<ul style="list-style-type: none"> <li>- Algılanan özellikleri korur</li> <li>- En çok kullanılan</li> </ul>	<ul style="list-style-type: none"> <li>- Boyut azaltarak örnekleme</li> <li>- LeNet'te kullanılmış</li> </ul>

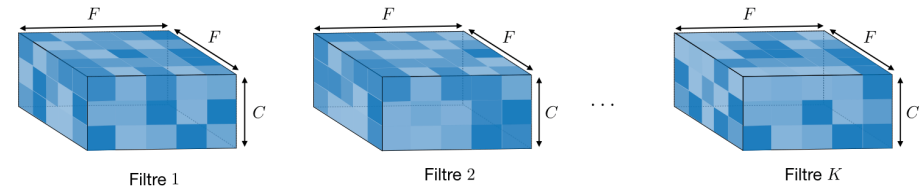
□ **Tam Bağlantı (FC)** – Tam bağlı katman (FC), her girişin tüm nöronlara bağlı olduğu bir giriş üzerinde çalışır. Eğer varsa, FC katmanları genellikle CNN mimarisinin sonuna doğru bulunur ve sınıf skorları gibi hedefleri optimize etmek için kullanılabilir.



## Hiperparametrelerin filtrelenmesi

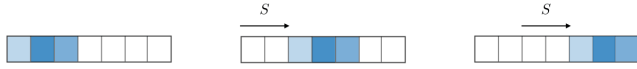
Evrişim katmanı, hiperparametrelerinin ardındaki anlamı bilmenin önemli olduğu filtreler içerir.

□ **Bir filtrenin boyutları** –  $C$  kanalları içeren bir girişe uygulanan  $F \times F$  boyutunda bir filtre,  $I \times I \times C$  boyutundaki bir girişte evrişim gerçekleştiren ve aynı zamanda bir çıkış öznetelik haritası üretir ( $O$  aktivasyon olarak da adlandırılır)  $O \times O \times 1$  boyutunda harita.



Not:  $F \times F$  boyutunda  $K$  filtrelerinin uygulanması,  $O \times O \times K$  boyutunda bir çıktı öznetelik haritasının oluşmasını sağlar.

□ **Adım aralığı** – Evrişimli veya bir ortaklama işlemi için,  $S$  adımı (adım aralığı), her işlemden sonra pencerenin hareket ettiği piksel sayısını belirtir.



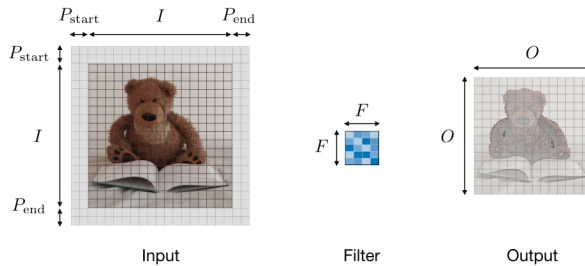
□ **Sıfır ekleme/doldurma** – Sıfır ekleme/doldurma, girişin sınırlarının her bir tarafına  $P$  sıfır ekleme işlemini belirtir. Bu değer manuel olarak belirlenebilir veya aşağıda detaylandırılan üç moddan biri ile otomatik olarak ayarlanabilir:

	Geçerli	Aynı	Tüm
Değer	$P = 0$	$P_{\text{start}} = \left\lfloor \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rfloor$ $P_{\text{end}} = \left\lceil \frac{S \lceil \frac{I}{S} \rceil - I + F - S}{2} \right\rceil$	$P_{\text{start}} \in [0, F - 1]$ $P_{\text{end}} = F - 1$
Görsel Açıklama			
Amaç	- Ekleme/doldurma yok - Boyutlar uyumuyorsa son evrişimi düşürür	- Öznitelik harita büyüklüğüne sahip ekleme/doldurma $\left\lceil \frac{I}{S} \right\rceil$ - Çıktı boyutu matematiksel olarak uygundur - 'Yarım' ekleme olarak da bilinir	- Son konvolüsyonların giriş sınırlarına uygulandığı maksimum ekleme - Filtre girişi uçtan uca 'görür'

## Hiperparametreleri ayarlama

□ **Evrişim katmanında parametre uyumu** – Girdinin hacim büyüklüğü  $I$  uzunluğu,  $F$  filtresinin uzunluğu,  $P$  sıfır ekleme miktarı,  $S$  adım aralığı, daha sonra bu boyut boyunca öznitelik haritasının  $O$  çıkış büyüklüğü belirtilir:

$$O = \frac{I - F + P_{\text{start}} + P_{\text{end}}}{S} + 1$$



Not: çoğunlukla,  $P_{\text{start}} = P_{\text{end}} \triangleq P$ , bu durumda  $P_{\text{start}} + P_{\text{end}}$ 'i yukarıdaki formülde  $2P$  ile değiştirebiliriz.

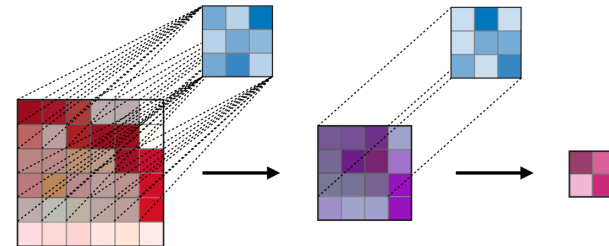
□ **Modelin karmaşıklığını anlama** – Bir modelin karmaşıklığını değerlendirmek için mimarisinin sahip olacağı parametrelerin sayısını belirlemek genellikle yararlıdır. Bir evrişimsel sinir ağının belirli bir katmanında, aşağıdaki şekilde yapılır:

	CONV	POOL	FC
Görsel Açıklama			
Giriş boyutu	$I \times I \times C$	$I \times I \times C$	$N_{\text{in}}$
Çıkış boyutu	$O \times O \times K$	$O \times O \times C$	$N_{\text{out}}$
Parametre sayısı	$(F \times F \times C + 1) \cdot K$	0	$(N_{\text{in}} + 1) \times N_{\text{out}}$
Not	- Filtre başına bir bias(önyargı) parametresi - Çoğu durumda, $S < F$ - $K$ için ortak bir seçenek $2C$ 'dir	- Ortaklama işlemi kanal bazında yapılır - Çoğu durumda, $S = F$	- Giriş bağlantılanmış - Nöron başına bir bias parametresi - Tam bağlantı (FC) nöronlarının sayısı yapısal kısıtlamalardan arındırılmış

□ **Evrişim sonucu oluşan haritanın boyutu** –  $K$  katmanında filtre çıkışı,  $k$ -inci aktivasyon haritasının her bir pikselinin 'görebileceği' girişin  $R_k \times R_k$  olarak belirtilen alanını ifade eder.  $F_j$ ,  $j$  ve  $S_i$  katmanlarının filtre boyutu,  $i$  katmanının adım aralığı ve  $S_0 = 1$  (ilk adım aralığının 1 seçilmesi durumu) kuralıyla,  $k$  katmanındaki işlem sonucunda elde edilen aktivasyon haritasının boyutları bu formülle hesaplanabilir:

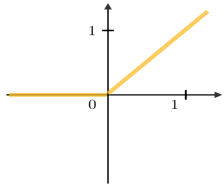
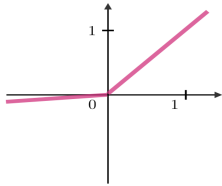
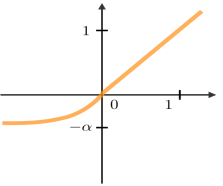
$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i$$

Aşağıdaki örnekte,  $F_1 = F_2 = 3$  ve  $S_1 = S_2 = 1$  için  $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$  sonucu elde edilir.



## Yaygın olarak kullanılan aktivasyon fonksiyonları

□ **Düzeltilmiş Doğrusal Birim** – Düzeltilmiş doğrusal birim katmanı (ReLU),  $(g)$ 'nin tüm elemanlarında kullanılan bir aktivasyon fonksiyonudur. Doğrusal olmamaları ile ağıın öğrenmesi amaçlanmaktadır. Çeşitleri aşağıdaki tabloda özetlenmiştir:


ReLU	Sızıntı ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ ile $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ ile $\alpha \ll 1$
		
Doğrusal olmama karmaşıklığı biyolojik olarak yorumlanabilir	Negatif değerler için ölen ReLU sorununu giderir	Her yerde türevlenebilir

□ **Softmax** – Softmax adımı,  $x \in \mathbb{R}^n$  skorlarının bir vektörünü girdi olarak alan ve mimarinin sonunda softmax fonksiyonundan  $p \in \mathbb{R}^n$  çıkış olasılık vektörünü oluşturan geliştirilmiş bir lojistik fonksiyon olarak görülebilir. Aşağıdaki gibi tanımlanır:

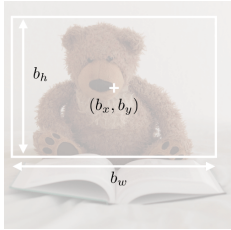
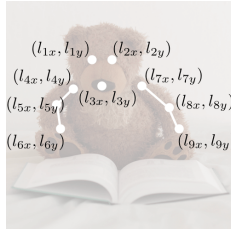
$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \quad \text{buna karşılık} \quad p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

## Nesne algılama

□ **Model tipleri** – Burada, nesne tanıma algoritmasının doğası gereği 3 farklı kestirim türü vardır. Aşağıdaki tabloda açıklanmıştır:

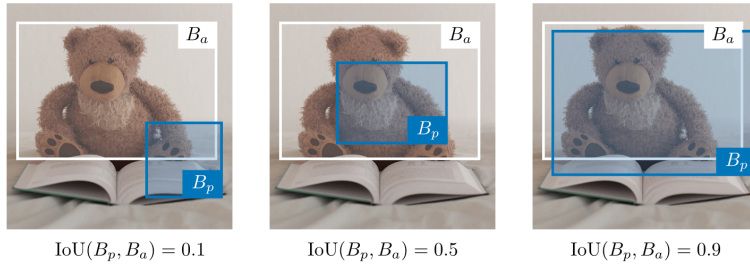
Görüntü sınıflandırma	Sınıflandırma ve lokalizasyon (konumlama)	Algılama
		
- Bir görüntüyü sınıflandırır - Nesnenin olasılığını tahmin eder	- Görüntüdeki bir nesneyi algılar/tanır - Nesnenin olasılığını ve bulunduğu yeri tahmin eder	- Bir görüntüdeki birden fazla nesneyi algılar - Nesnelerin olasılıklarını ve nerede olduklarını tahmin eder
Geleneksel CNN	Basitleştirilmiş YOLO, R-CNN	YOLO, R-CNN

□ **Algılama** – Nesne algılama bağlamında, nesneyi konumlandırmak veya görüntüdeki daha karmaşık bir şekli tespit etmek isteyip istemediğimize bağlı olarak farklı yöntemler kullanılır. İki ana tablo aşağıdaki tabloda özetlenmiştir:

Sınırlayıcı kutu ile tespit	Karakteristik nokta algılama
Görüntüde nesnenin bulunduğu yeri algılar	- Bir nesnenin şeklini veya özelliklerini algılar (örneğin gözler) - Daha ayrıntılı
	
Kutu merkezi $(b_x, b_y)$ , yükseklik $b_h$ ve genişlik $b_w$	Referans noktalar $(l_{1x}, l_{1y}), \dots, (l_{nx}, l_{ny})$

□ **Kesiştirilmiş Bölgeler** – Kesiştirilmiş Bölgeler, IoU (*Intersection over Union*) olarak da bilinir, Birleştirilmiş sınırlama kutusu, tahmin edilen sınırlama kutusu ( $B_p$ ) ile gerçek sınırlama kutusu  $B_a$  üzerinde ne kadar doğru konumlandırıldığını ölçen bir fonksiyondur. Olarak tanımlanır:

$$\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}$$

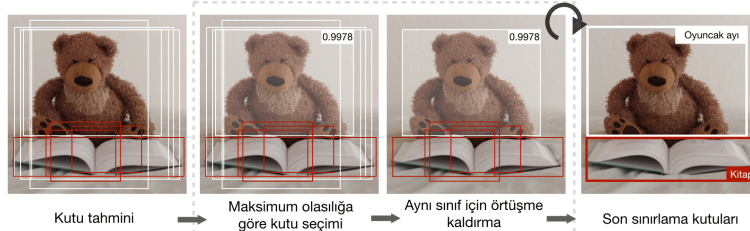


*Not: Her zaman  $\text{IoU} \in [0,1]$  ile başlarız. Kural olarak, Öngörülen bir sınırlama kutusu  $B_p$ ,  $\text{IoU}(B_p, B_a) \geq 0.5$  olması durumunda makul derecede iyi olarak kabul edilir.*

□ **Öneri kutular** – Öneri (Anchor) kutular, örtüşen sınırlayıcı kutuları öngörmek için kullanılan bir tekniktir. Uygulamada, ağı aynı anda birden fazla kutuyu tahmin etmesine izin verilir, burada her kutu tahmini belirli bir geometrik öznitelik setine sahip olmakla sınırlıdır. Örneğin, ilk tahmin potansiyel olarak verilen bir formun dikdörtgen bir kutusudur, ikincisi ise farklı bir geometrik formun başka bir dikdörtgen kutusudur.

□ **Maksimum olmayan bastırma** – Maksimum olmayan bastırma tekniği, nesne için yinelenen ve örtüşen öneri kutuları içinde en uygun temsilleri seçerek örtüşmesi düşük olan kutuları kaldırmayı amaçlar. Olasılık tahmini 0.6'dan daha düşük olan tüm kutuları çıkardıktan sonra, kalan kutular ile aşağıdaki adımlar tekrarlanır: Verilen bir sınıf için,

- Adım 1: En büyük tahmin olasılığı olan kutuyu seçin.
- Adım 2: Önceki kutuyla  $\text{IoU} \geq 0.5$  olan herhangi bir kutuyu çıkarın.



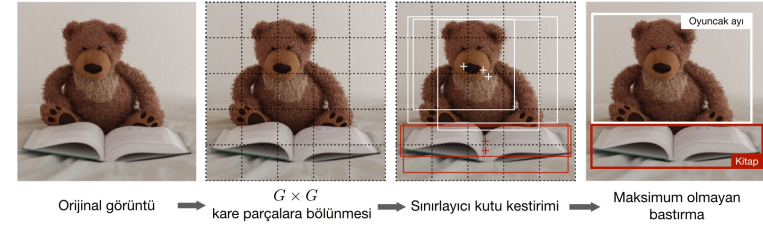
□ **YOLO** – You Only Look Once (YOLO), aşağıdaki adımları uygulayan bir nesne algılama algoritmasıdır:

- Adım 1: Giriş görüntüsünü  $G \times G$  kare parçalara (hücrelere) bölün.
- Adım 2: Her bir hücre için, aşağıdaki formdan  $y$ 'yi öngören bir CNN çalıştırın:

$$y = \left[ \underbrace{p_c, b_x, b_y, b_h, b_w, c_1, c_2, \dots, c_p}_{k \text{ kez tekrarlayan}} \right]^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

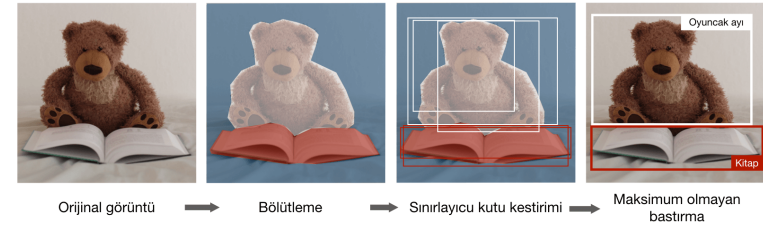
$p_c$ 'nin bir nesneyi algılama olasılığı olduğu durumlarda,  $b_x, b_y, b_h, b_w$  tespit edilen olası sınırlayıcı kutusunun özellikleridir,  $c_1, \dots, c_p$ ,  $p$  sınıflarının tespit edilen one-hot temsildir ve  $k$  öneri ( $\langle i \rangle$  anchor) kutularının sayısıdır.

- Adım 3: Potansiyel yineli çıkan sınırlayıcı kutuları kaldırmak için maksimum olmayan bastırma algoritmasını çalıştır.



*Not:  $p_c = 0$  olduğunda, ağ herhangi bir nesne algılamamaktadır. Bu durumda, ilgili  $b_x, \dots, c_p$  tahminleri dikkate alınmamalıdır.*

□ **R-CNN** – Evrişimli Sinir Ağları ile Bölge Bulma (R-CNN), potansiyel olarak sınırlayıcı kutuları bulmak için görüntüyü bölütleyen (segmente eden) ve daha sonra sınırlayıcı kutularda en olası nesneleri bulmak için algılama algoritmasını çalıştıran bir nesne algılama algoritmasıdır.



*Not: Orijinal algoritma hesaplamalı olarak maliyetli ve yavaş olmasına rağmen, yeni mimariler algoritmanın Hızlı R-CNN ve Daha Hızlı R-CNN gibi daha hızlı çalışmasını sağlamıştır.*

## Yüz doğrulama ve tanıma

□ **Model tipleri** – İki temel model aşağıdaki tabloda özetlenmiştir:

Yüz doğrulama	Yüz tanıma
<ul style="list-style-type: none"> <li>- Bu doğru kişi mi?</li> <li>- Bire bir arama</li> </ul>	<ul style="list-style-type: none"> <li>- Veritabanındaki <math>K</math> kişilerden biri mi?</li> <li>- Bire-çok arama</li> </ul>
<p>Sorgu</p> <p>Kaynak</p>	<p>Sorgu</p> <p>Veri tabanı</p>

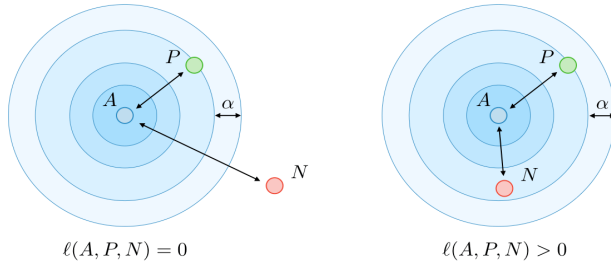
□ **Tek Atış (One-Shot) Öğrenme** – Tek Atış Öğrenme, verilen iki görüntünün ne kadar farklı olduğunu belirleyen benzerlik fonksiyonunu öğrenmek için sınırlı bir eğitim seti kullanan bir yüz doğrulama algoritmasıdır. İki resme uygulanan benzerlik fonksiyonu sıklıkla kaydedilir  $d(\text{resim 1}, \text{resim 2})$ .



□ **Siyam (Siamese) Ağı** – Siyam Ağı, iki görüntünün ne kadar farklı olduğunu ölçmek için görüntülerin nasıl kodlanacağını öğrenmeyi amaçlar. Belirli bir giriş görüntüsü  $x^{(i)}$  için kodlanmış çıkış genellikle  $f(x^{(i)})$  olarak alınır.

□ **Üçlü kayıp** – Üçlü kayıp  $\ell$ ,  $A$  (öneri),  $P$  (pozitif) ve  $N$  (negatif) görüntülerinin üçlüsünün gömülü gösterimde hesaplanan bir kayıp fonksiyonudur. Öneri ve pozitif örnek aynı sınıfa aitken, negatif örnek bir diğere aittir.  $\alpha \in \mathbb{R}^+$  marjın parametresini çağırarak, bu kayıp aşağıdaki gibi tanımlanır:

$$\ell(A, P, N) = \max(d(A, P) - d(A, N) + \alpha, 0)$$



### Sinirsel stil transferi (aktarımı)

□ **Motivasyon** – Sinirsel stil transferinin amacı, verilen bir  $C$  içeriğine ve verilen bir  $S$  stile dayanan bir  $G$  görüntüsü oluşturmaktır.



□ **Aktivasyon** – Belirli bir  $l$  katmanında, aktivasyon  $a^{[l]}$  olarak gösterilir ve  $n_H \times n_w \times n_c$  boyutlarındadır.

□ **İçerik maliyeti fonksiyonu** – İçerik maliyeti fonksiyonu  $J_{\text{content}}(C, G)$ ,  $G$  oluşturulan görüntüsünün,  $C$  orijinal içerik görüntüsünden ne kadar farklı olduğunu belirlemek için kullanılır. Aşağıdaki gibi tanımlanır:

$$J_{\text{content}}(C, G) = \frac{1}{2} \|a^{[l](C)} - a^{[l](G)}\|^2$$

□ **Stil matrisi** – Stil matrisi  $G^{[l]}$ , belirli bir  $l$  katmanının her birinin  $G_{kk'}^{[l]}$  elemanlarının  $k$  ve  $k'$  kanallarının ne kadar ilişkili olduğunu belirlediği bir Gram matristir.  $a^{[l]}$  aktivasyonlarına göre aşağıdaki gibi tanımlanır:

$$G_{kk'}^{[l]} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_w} a_{ijk}^{[l]} a_{ijk'}^{[l]}$$

*Not: Stil görüntüsü ve oluşturulan görüntü için stil matrisi, sırasıyla  $G^{[l](S)}$  ve  $G^{[l](G)}$  olarak belirtilmiştir.*

□ **Stil maliyeti fonksiyonu** – Stil maliyeti fonksiyonu  $J_{\text{style}}(S, G)$ , oluşturulan  $G$  görüntüsünün  $S$  stilinden ne kadar farklı olduğunu belirlemek için kullanılır. Aşağıdaki gibi tanımlanır:

$$J_{\text{style}}^{[l]}(S, G) = \frac{1}{(2n_H n_w n_c)^2} \|G^{[l](S)} - G^{[l](G)}\|_F^2 = \frac{1}{(2n_H n_w n_c)^2} \sum_{k,k'=1}^{n_c} \left( G_{kk'}^{[l](S)} - G_{kk'}^{[l](G)} \right)^2$$

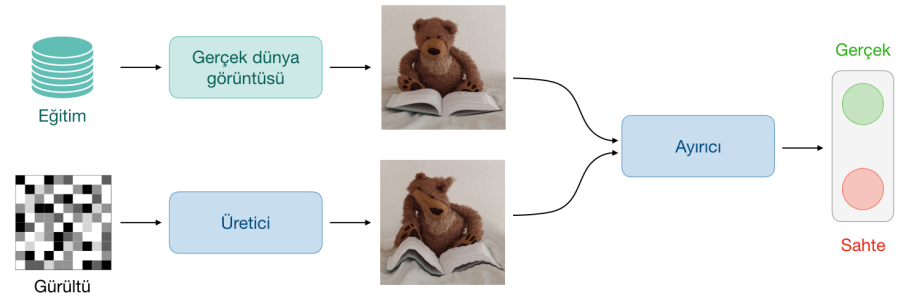
□ **Genel maliyet fonksiyonu** – Genel maliyet fonksiyonu,  $\alpha, \beta$  parametreleriyle ağırlıklandırılan içerik ve stil maliyet fonksiyonlarının bir kombinasyonu olarak tanımlanır:

$$J(G) = \alpha J_{\text{content}}(C, G) + \beta J_{\text{style}}(S, G)$$

*Not: yüksek bir  $\alpha$  değeri modelin içeriğe daha fazla önem vermesini sağlarken, yüksek bir  $\beta$  değeri de stile önem verir.*

### Hesaplama ipuçları kullanan mimariler

□ **Çekişmeli Üretici Ağlar** – GAN olarak da bilinen çekişmeli üretici ağlar, modelin üretici denen ve gerçek imajı ayırt etmeyi amaçlayan ayırtıcıya beslenecek en doğru çıktının oluşturulmasını amaçladığı üretici ve ayırt edici bir modelden oluşur.



*Not: GAN'ın kullanım alanları, yazıdan görüntüye, müzik üretimi ve sentezi.*

□ **ResNet** – Artık Ağ mimarisi (ResNet olarak da bilinir), eğitim hatasını azaltmak için çok sayıda katman içeren artık bloklar kullanır. Artık blok aşağıdaki karakterizasyon denklemine sahiptir:

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

□ **Inception Ağ** – Bu mimari inception modüllerini kullanır ve özelliklerini çeşitlendirme yoluyla performansını artırmak için farklı evrişim kombinasyonları denemeyi amaçlamaktadır. Özellikle, hesaplama yükünü sınırlamak için 1x1 evrişim hilesini kullanır.

★ ★ ★