ANKARA ÜNİVERSİTESİ
SİBER GÜVENLİK MESLEK YÜKSEKOKULU

Çift Modüllü Ağlar ile Kötü Yazılım (Malware) Açık-küme Tanıma

Multimodal Dual-Embedding Networks for Malware Open-Set Recognition

PROBLEM ve MOTİVASYON

- Bu çalışmada, malware (kötü amaçlı yazılım) sınıflandırması yapılmıştır.
- Malware'ler sürekli olarak gelişmekte ve yeni türler ortaya çıkmaktadır. Bu durum, malware tespiti ve sınıflandırması için modellerin yalnızca bilinen malware türlerini değil, aynı zamanda bilinmeyen veya yeni ortaya çıkan türleri de tanıyabilmesini gerektirmektedir.
- Bu bağlamda, "open-set recognition" (açık küme tanıma) problemi ele alınmıştır.

OPEN-SET RECOGNITION (AÇIK KÜME TANIMA)

- Open-set recognition, bir modelin hem bilinen sınıfları doğru bir şekilde sınıflandırması hem de eğitim verisinde bulunmayan yeni veya bilinmeyen sınıfları tanıyabilmesi problemidir.

- Gerçek dünyada sürekli yeni malware türleri ortaya çıktığından, bir malware tespit sisteminin hem bilinen tehditleri sınıflandırabilmesi hem de yeni tehditleri tanıyabilmesi gerekmektedir.

OPEN-SET RECOGNITION (AÇIK KÜME TANIMA)

- **Eşik Temelli Yöntemler:** Sınıflandırma güven skoru üzerinde bir eşik belirlenir. Eğer en yüksek güven skoru belirli bir eşik değerinin altındaysa, örnek bilinmeyen olarak sınıflandırılır.

- **Mesafe Temelli Yöntemler:** Girdi örneğinin özellik temsilinin bilinen sınıf merkezlerine olan mesafesi ölçülür. Eğer mesafe belirli bir eşik değerinden büyükse, örnek bilinmeyen olarak kabul edilir.

VERİ SETİ

- Bu çalışmada, önerilen yöntemin test edilmesi ve değerlendirilmesi için MAL-100+ veri seti kullanılmıştır.

- Bu veri seti, 100 adet malware türlerini içermektedir ve hem statik hem de dinamik analiz bilgilerini barındırmaktadır. Buna ek olarak etiketlendirilmiş 50,000 adet görüntü bulunmaktadır.

ÖZGÜNLÜK

- Çalışmanın özgünlüğü bilinmeyen kötü yazılımların da sınıflandırılması olarak belirtilmiş.

- Çünkü bilinen (hatta bilinmeyen) kötü yazılımlar birbirine benzer özelliklere sahip olduğundan literatürdeki sınıflandırma teknikleri bu kötü yazılımları sınıflandırmada iyi olarak gözükebilir ama bilinmeyen kötü yazılımların da özellikleri bilinen kötü yazılımlara benzer olduğundan sınıflandırma işlemi yanlış olabilir. Model doğru olduğunu söylese de…

ÖZGÜNLÜK

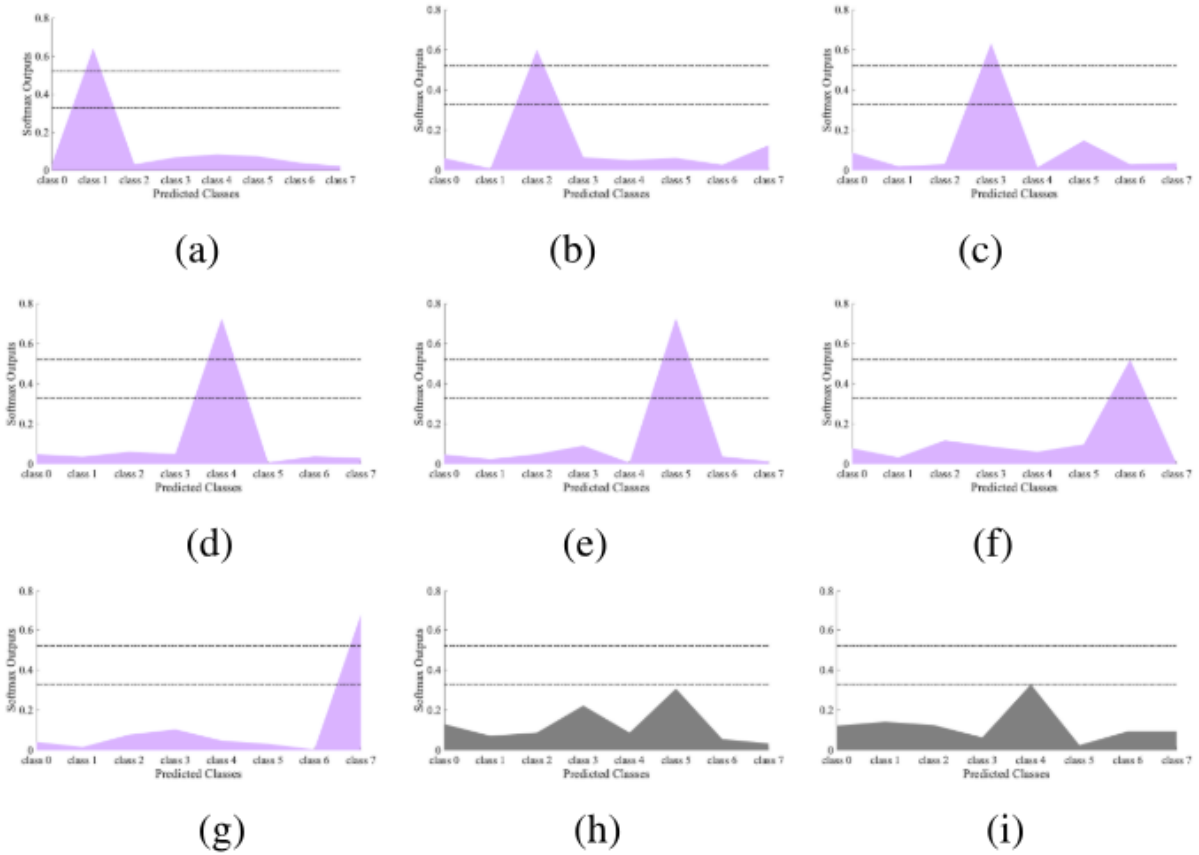


Fig. 1. Visualization of softmax outputs of digits on known classes "1"–"7" and unknown classes "8" and "9." (a)–(g) Marked in violet are known classes and (h) and (i) marked in gray are unknown classes (better viewed in color) (a) Digit 1. (b) Digit 2. (c) Digit 3. (d) Digit 4. (e) Digit 5. (f) Digit 6 (g) Digit 7. (h) Digit 8 (unknown). (i) Digit 9 (unknown).

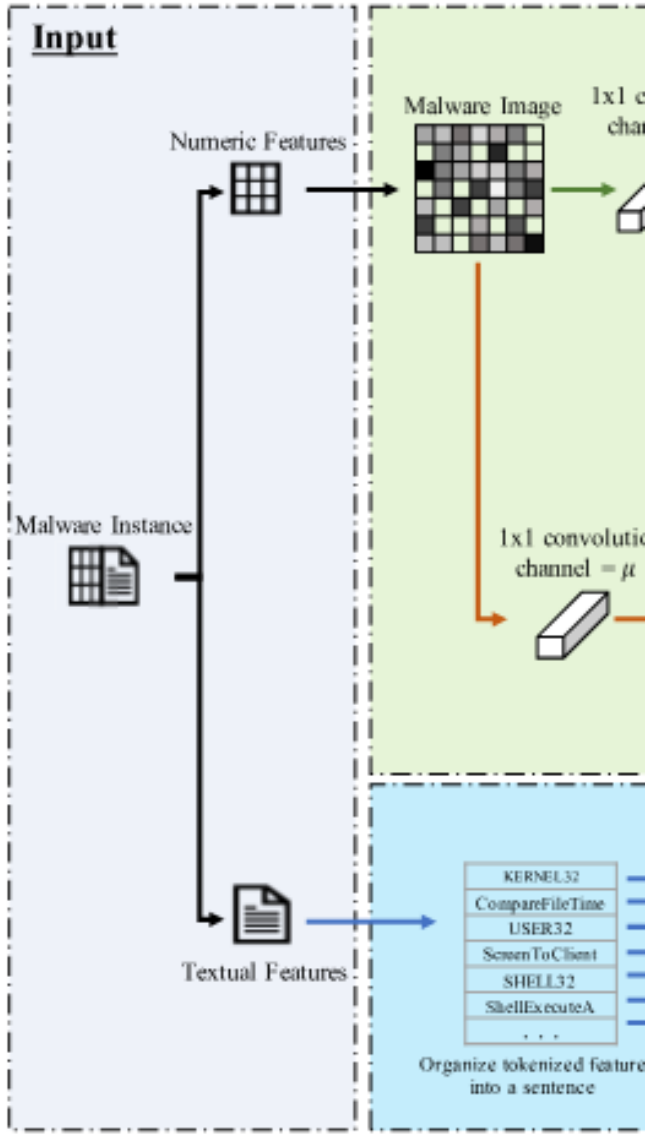
KÖTÜ YAZILIM ÖZELLİKLERİ

- Çalışmada kötü yazılımların özellikleri text ve nümerik olmak üzere ikiye ayrılmış.
- Ember adlı bir yazılım kullanılmış.
- Nümerik özellikler: Dosya boyutu, n-bitlik parçaları, entropi değeri vs.
- Text özellikleri: API çağrıları (Readfile, Createfile vs.), sistem kayıtlarına yapılan değişiklikler, bellek alanlarına okuma yazma işlemleri vs.

KÖTÜ YAZILIM ÖZELLİKLERİ

- Elde edilen nümerik özelliklerden malware görüntüsü oluşturulmuş.

- Elde edilen text özelliklerinden ise malware cümleleri oluşturulmuş.



ÖNERİLEN YÖNTEM

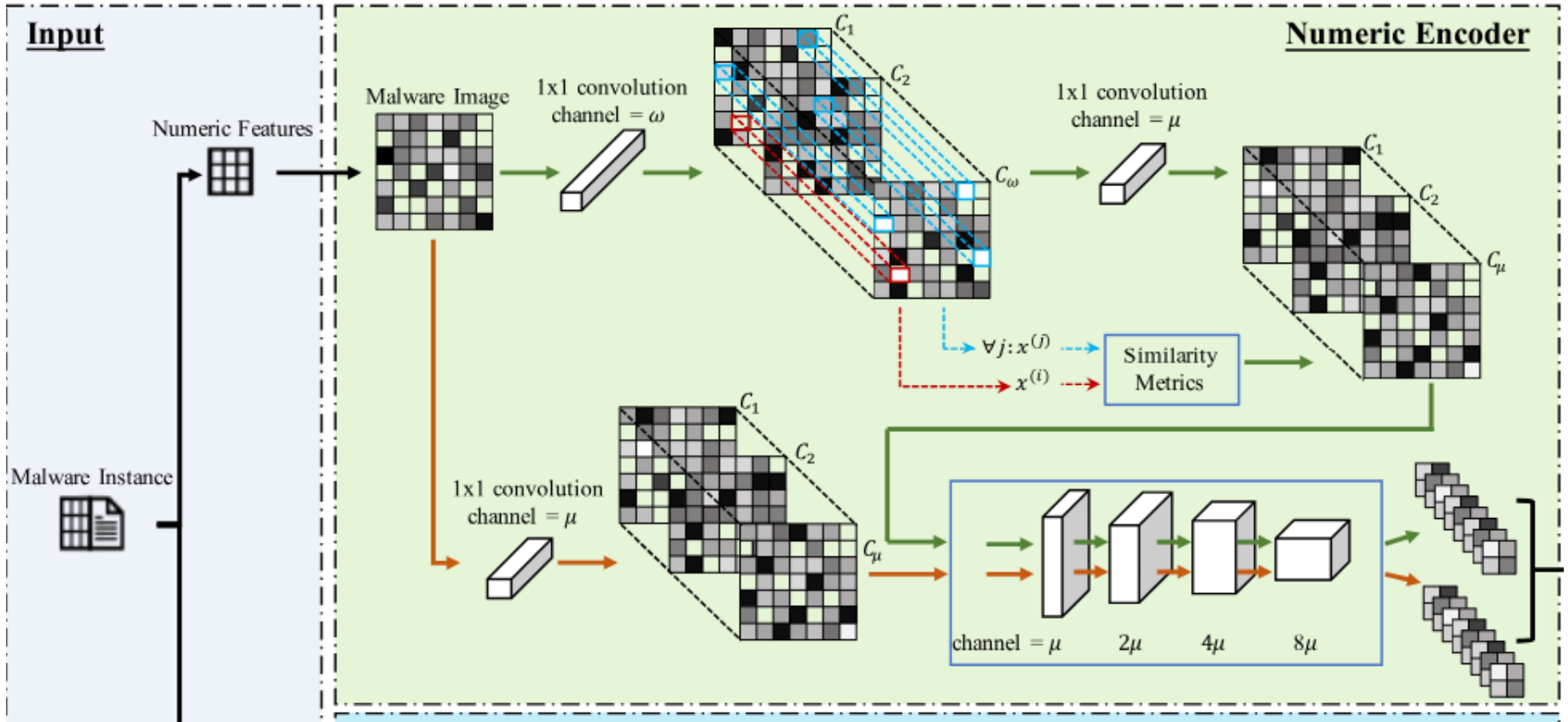
- Malware görüntülerin özelliklerini öğrenebilmek için evrişimsel sinir ağları, malware cümlelerin özelliklerini öğrenebilmek için BERT modeli kullanılmış.

1. Evrişimsel Sinir Ağları: Görüntü tanıma ve analizinde kullanılan derin öğrenme modelidir.

2. BERT: Büyük miktarda metin verisiyle önceden eğitilmiş bir tür yapay zeka modelidir. Metinlerdeki kelimeler arasındaki ilişkileri daha iyi anlamak için çalışır. Özellikle metin sınıflandırma, soru-cevap gibi görevlerde kullanılır.

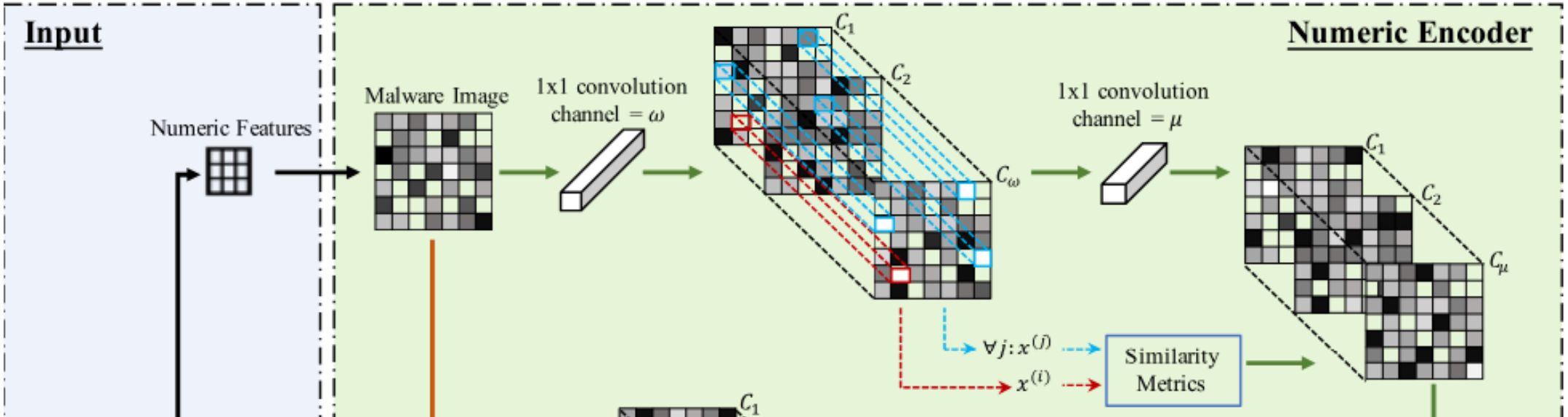
ÖNERİLEN YÖNTEM (ESA)

- ESA kısmı da iki parçaya ayrılmış: global receptive module, local receptive module



ÖNERİLEN YÖNTEM (ESA: Global Receptive Module)

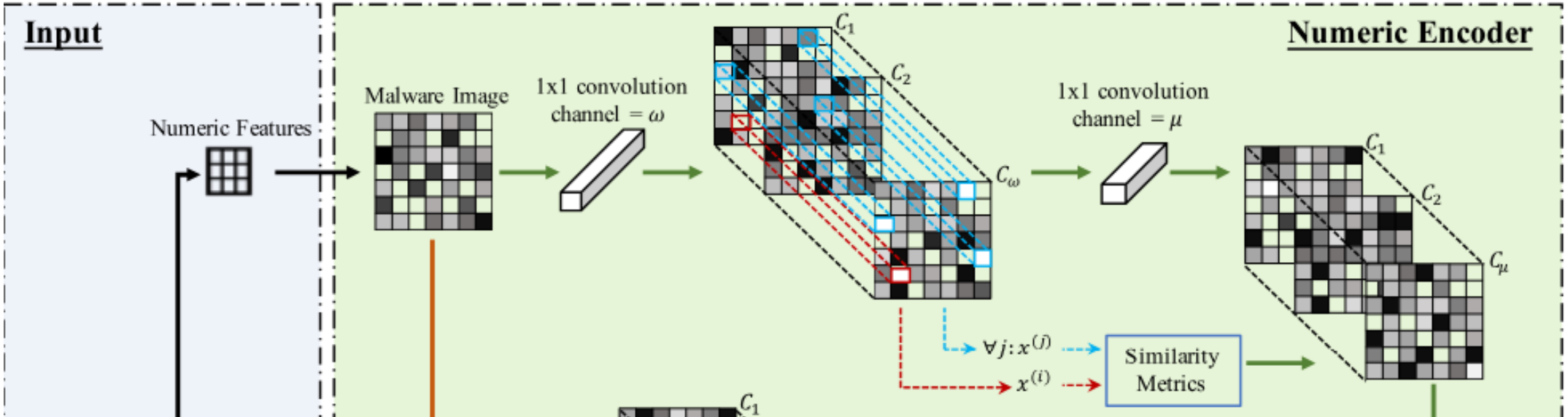
- 128 ve 64 kanal ile iki adet 1×1 konvolüsyon katmanı.
- Özellik haritalarında noktalar arasında benzerlik ölçümü (Gaussian fonksiyonu ile).
- Bu modülün amacı uzak-mesafe özelliklerinin yakalanması için.



ÖNERİLEN YÖNTEM (ESA: Global Receptive Module)

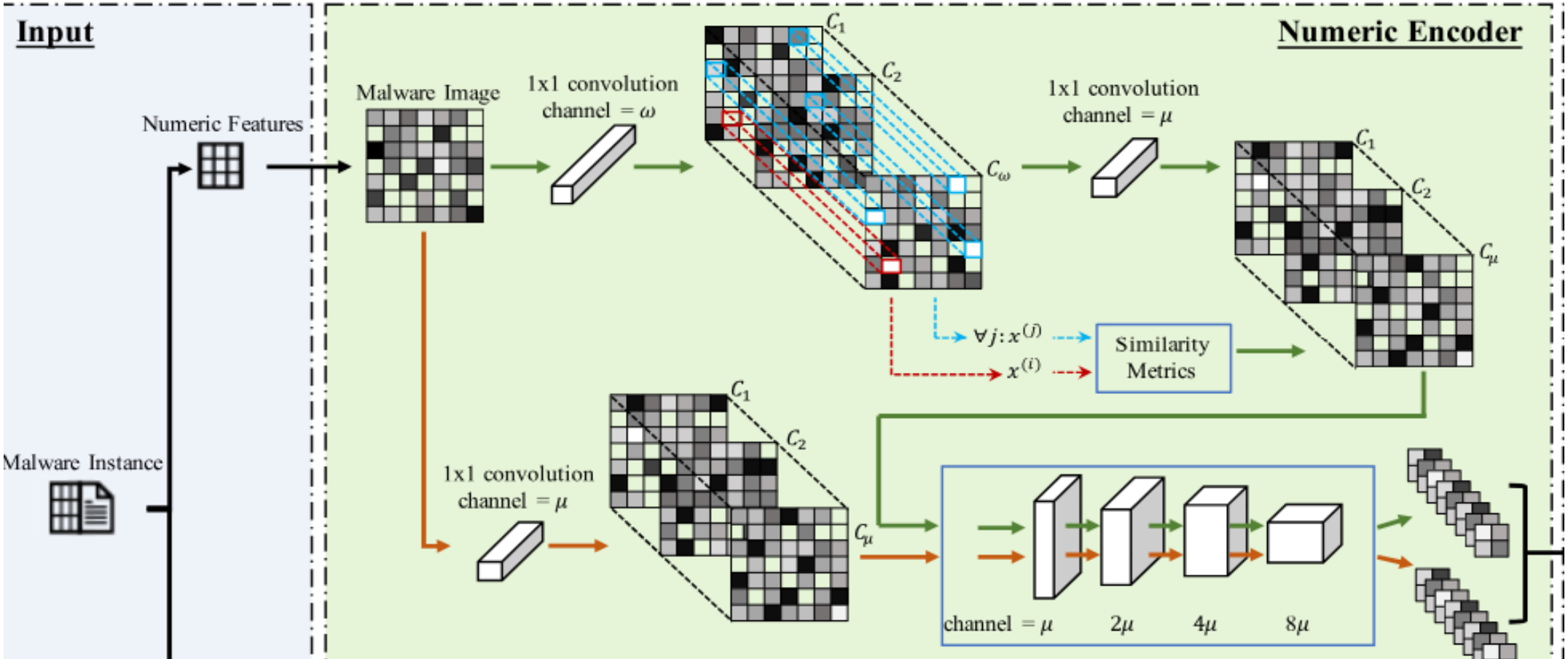
- Tipik bir CNN mimarisi, bir görüntüde bir pikselin ve yakındaki piksellerin bağımlılıklarını yakalamaya odaklanır.

- Ancak, kötü amaçlı yazılım özellikleri genellikle gerçek görüntülerde olduğu gibi güçlü bir lokaliteye sahip olmayabilir. Bu nedenle, kötü amaçlı yazılım örnekleri genellikle özellikler arasında çok ölçekli ilişkilere sahiptir.



ÖNERİLEN YÖNTEM (ESA: Local Receptive Module)

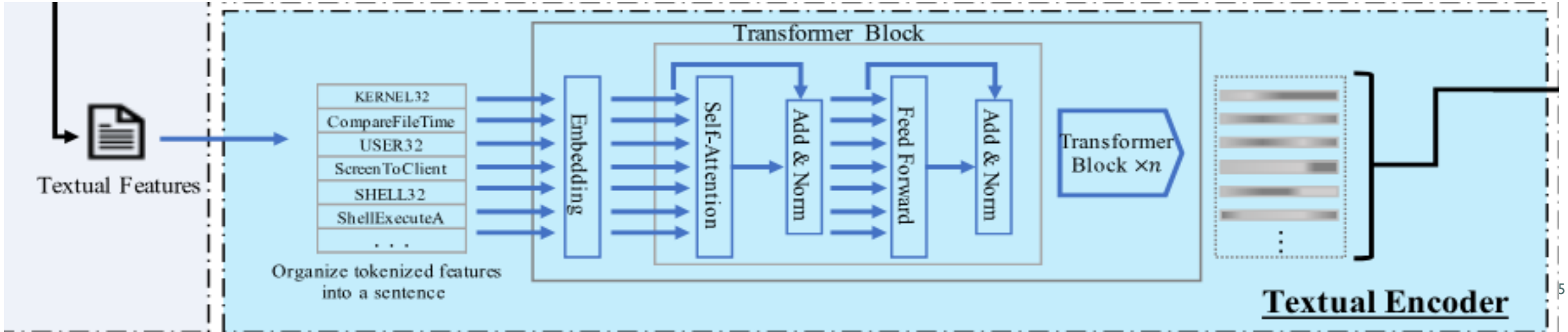
- Tipik bir CNN mimarisi, bir görüntüde bir pikselin ve yakındaki piksellerin bağımlılıklarını yakalamaya odaklanır.



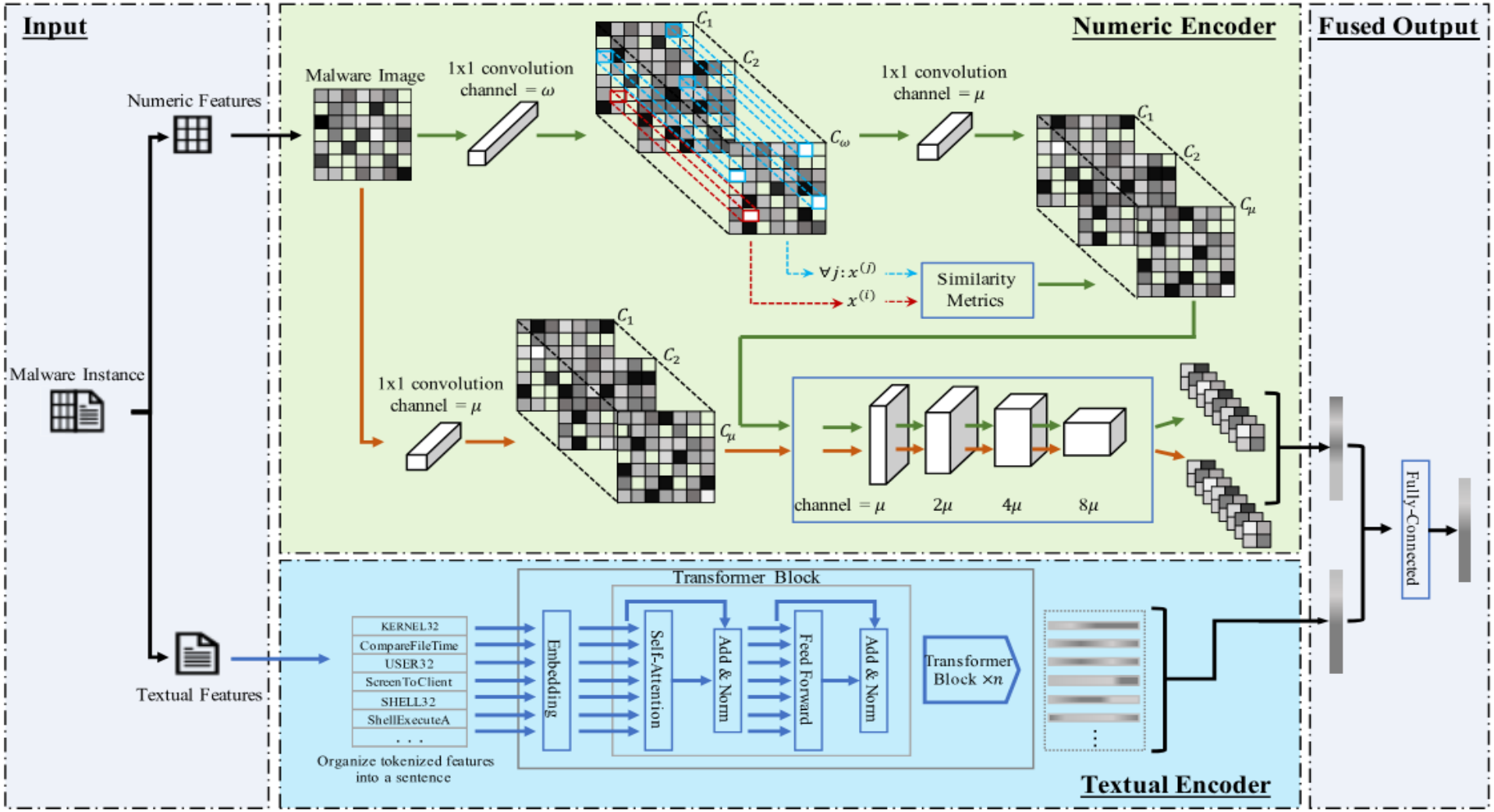
ÖNERİLEN YÖNTEM (BERT)

- Geleneksel yöntemlerden farklı olarak, her kötü amaçlı yazılım örneğinin tokenleştirilmiş metin özellikleri bir cümleye düzenlenir. Bu cümlede, kelimeler açıkça bazı önemli alanları temsil eder; bu alanlar kötü amaçlı yazılım davranışları ve dinamiklerini içerir.

- BERT'in çıktısı, metnin her kelimesini temsil eden nümerik vektörlerden oluşur.



ÖNERİLEN YÖNTEM



HATA FONKSİYONU

- Yapay Zeka Modellerinin eğitiminde bir hata fonksiyonunun olması gerekir. Klasik sınıflandırma modellerinde aşağıdaki hata fonksiyonu kullanılır.

$$\mathrm{Loss}_{\mathrm{Cls}}(\Theta_{\mathrm{E}}, \Theta_{\mathrm{C}}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \sum_{k=1}^{K} -p_k(l_i) \log q_k(x_i, s_i)$$

- Fakat bu hata fonksiyonu close-set, yani veri setinde tanımlanmış sınıfların sınıflandırılması içindir.

HATA FONKSİYONU

- Yazarların amacı sadece bilinen malware sınıflarının değil, bilinmeyen malware sınıflarının da tespiti olduğu için kayıp fonksiyonuna iki ekleme daha yapmışlardır.

HATA FONKSİYONU (Birinci Ekleme)

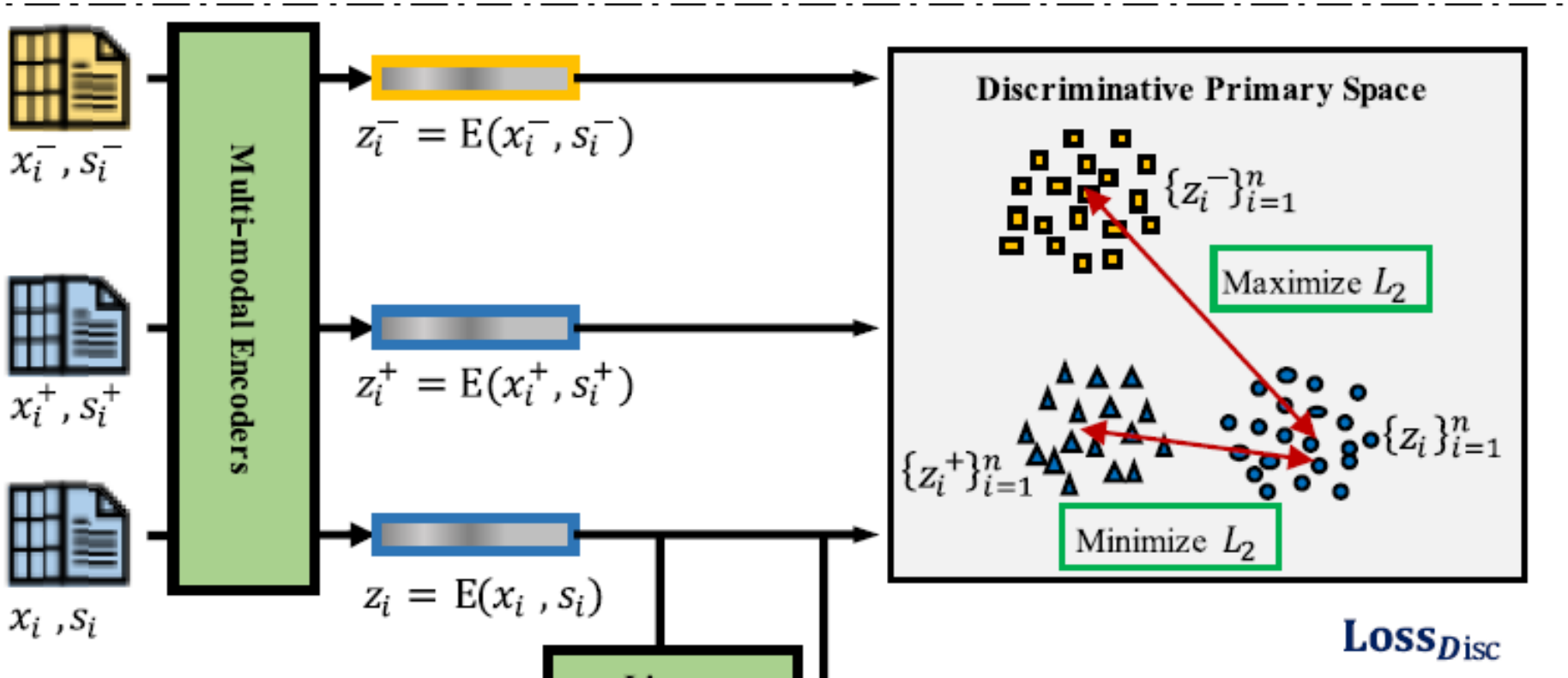
- Her bir örnek (x_i: Malware Image , s_i: Malware Sentence) için rastgele iki örnek çifti seçilir (bir iterasyon için). Bu örnek çiftleri şu şekildedir:

1. (x^{+i}, s^{+i}): Bu çift, orijinal örneğin (x_i, s_i) sınıfı ile aynı sınıfa ait herhangi bir örnektir.

2. (x^{-i}, s^{-i}): Bu çift, orijinal örneğin (x_i, s_i) sınıfı ile farklı sınıfa ait herhangi bir örnektir.

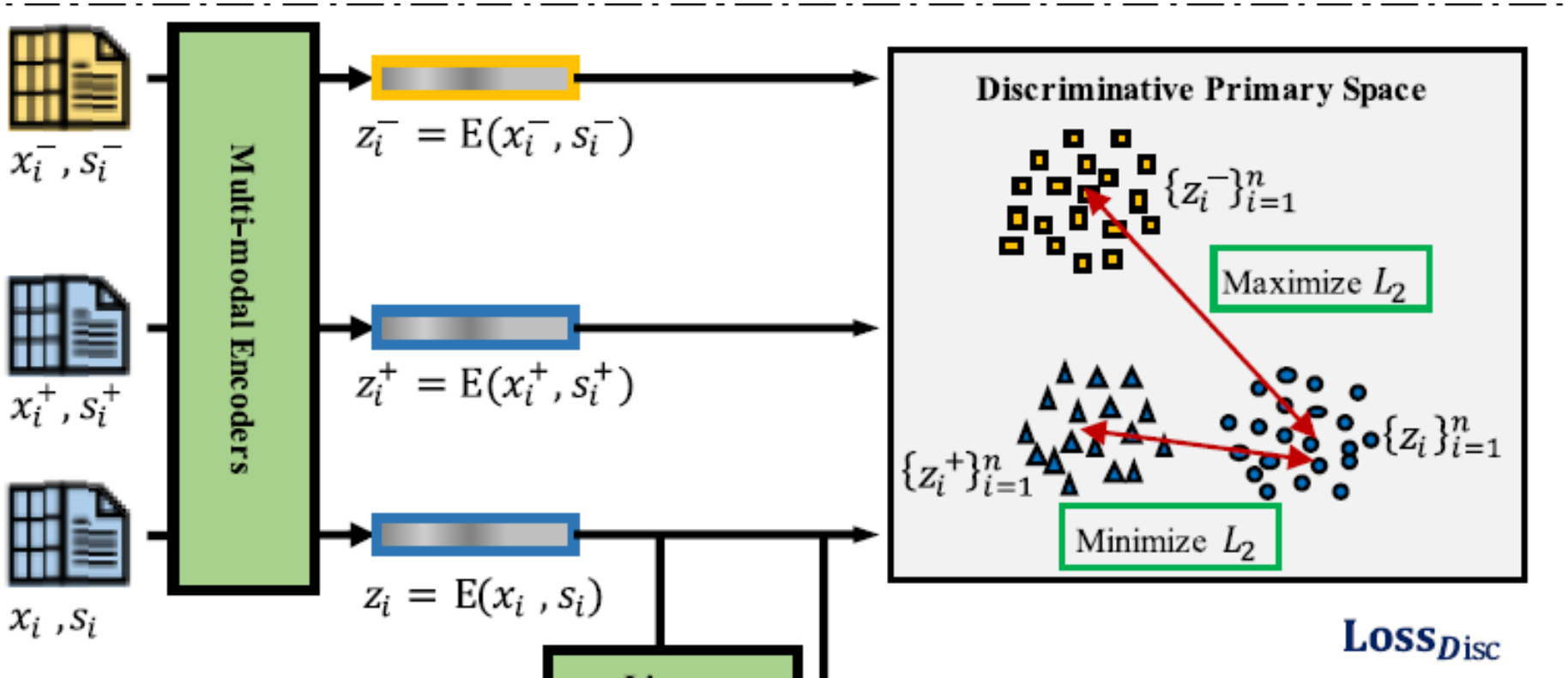
HATA FONKSİYONU (Birinci Ekleme)

- İki örnek çifti seçildikten sonra, orijinal çift ve bu iki örnek çift yapay zeka modeline gönderildikten sonra her biri için özellikler (modelin çıktısı) elde edilir. Ardından bu özellikler bir örnek uzayına gönderilir.



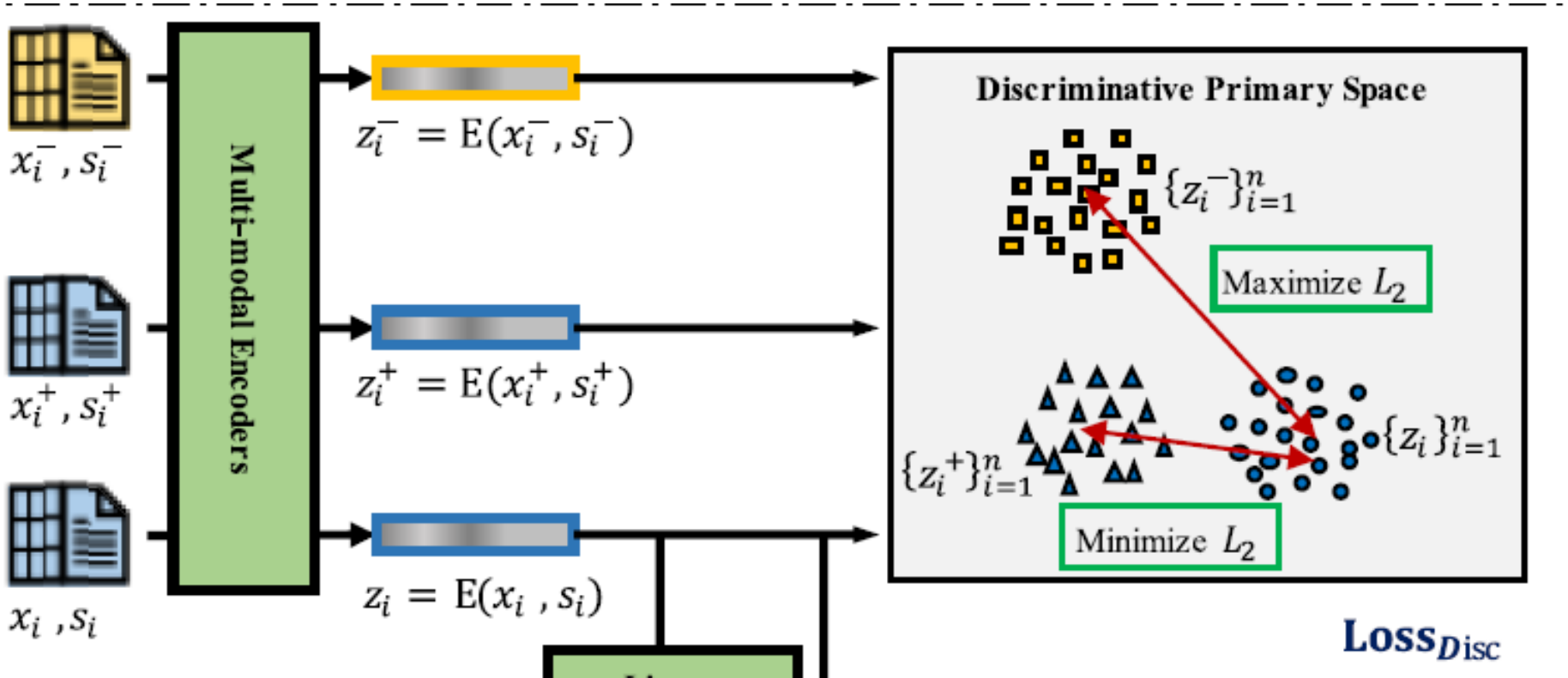
HATA FONKSİYONU (Birinci Ekleme)

- Burada maviler aynı sınıfa, sarı ise farklı sınıfa ait bir örnektir. Buradaki amaç aynı sınıfa ait özelliklerin birbirine yakınlaştırılıp, farklı sınıfa ait özelliklerin birbirinden uzaklaştırılması sağlanır (L_2 Normalization).



HATA FONKSİYONU (Birinci Ekleme)

- Kısacası modele, aynı sınıfa ait özellikler birbirinden farka sahip, o özellikleri yakınlaştır veya farklı sınıfa ait özellikler birbirine yakın o yüzden o özellikleri uzaklaştır denir.



HATA FONKSİYONU (Birinci Ekleme)

- Bu hata fonksiyonu modelin sınıflandırma başarısını artırmak için bir fonksiyon, ayrıca bilinmeyen sınıfların sınıflandırması için de bir ön hazırlıktır.

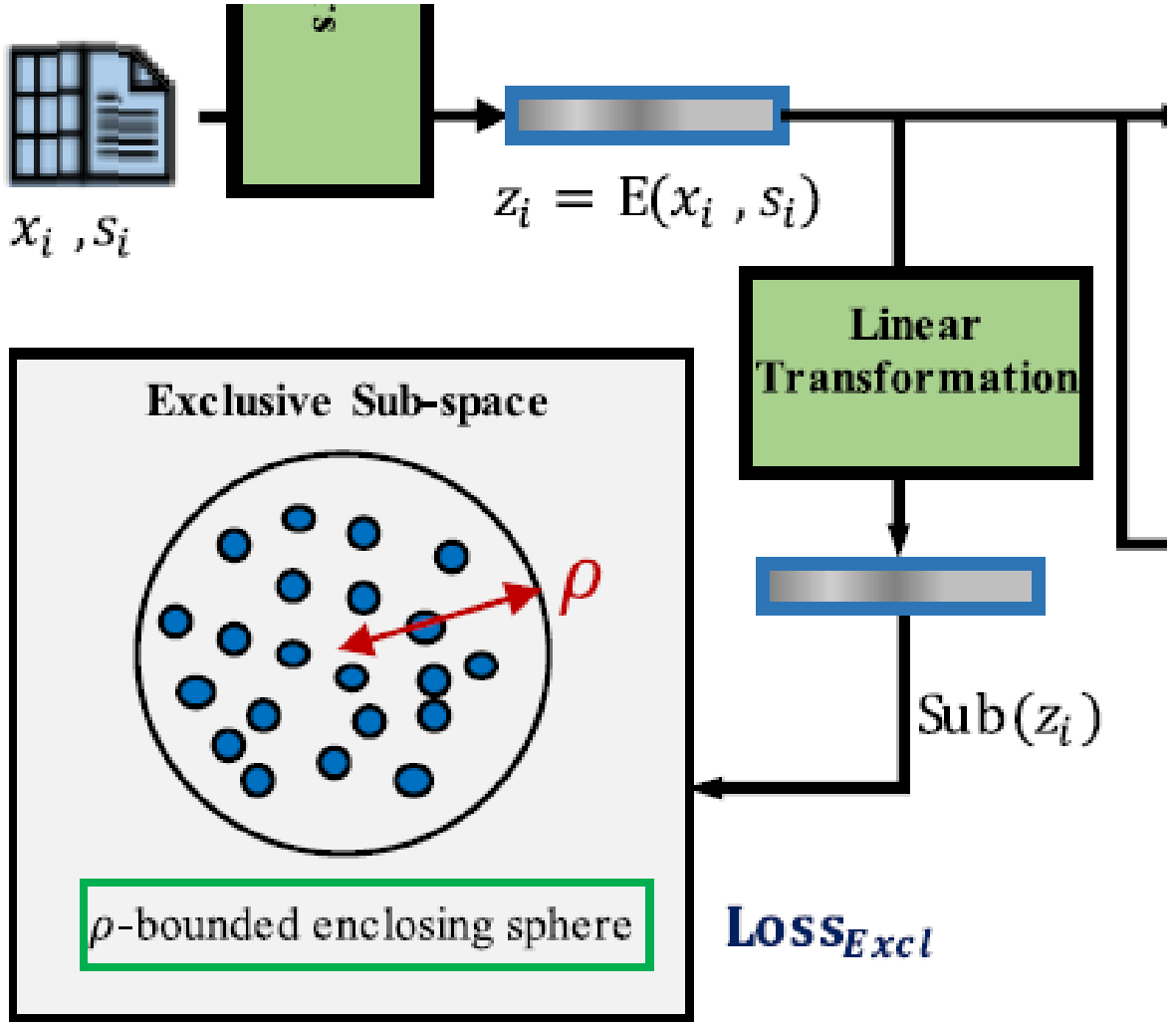
$$\mathrm{Loss}_{\mathrm{Disk}}(\Theta_{\mathrm{E}}) = \sum_{i=1}^{n} \left(\|z_i - z_i^{+}\|_2 - \|z_i - z_i^{-}\|_2 \right).$$

HATA FONKSİYONU (İkinci Ekleme)

- Birinci eklemeye rağmen, model hala bilinmeyen malware sınıflarının tespitini yapacak seviyede değildir. Bu yüzden yazarlar ikinci bir ekleme yapmışlardır.

HATA FONKSİYONU (İkinci Ekleme)

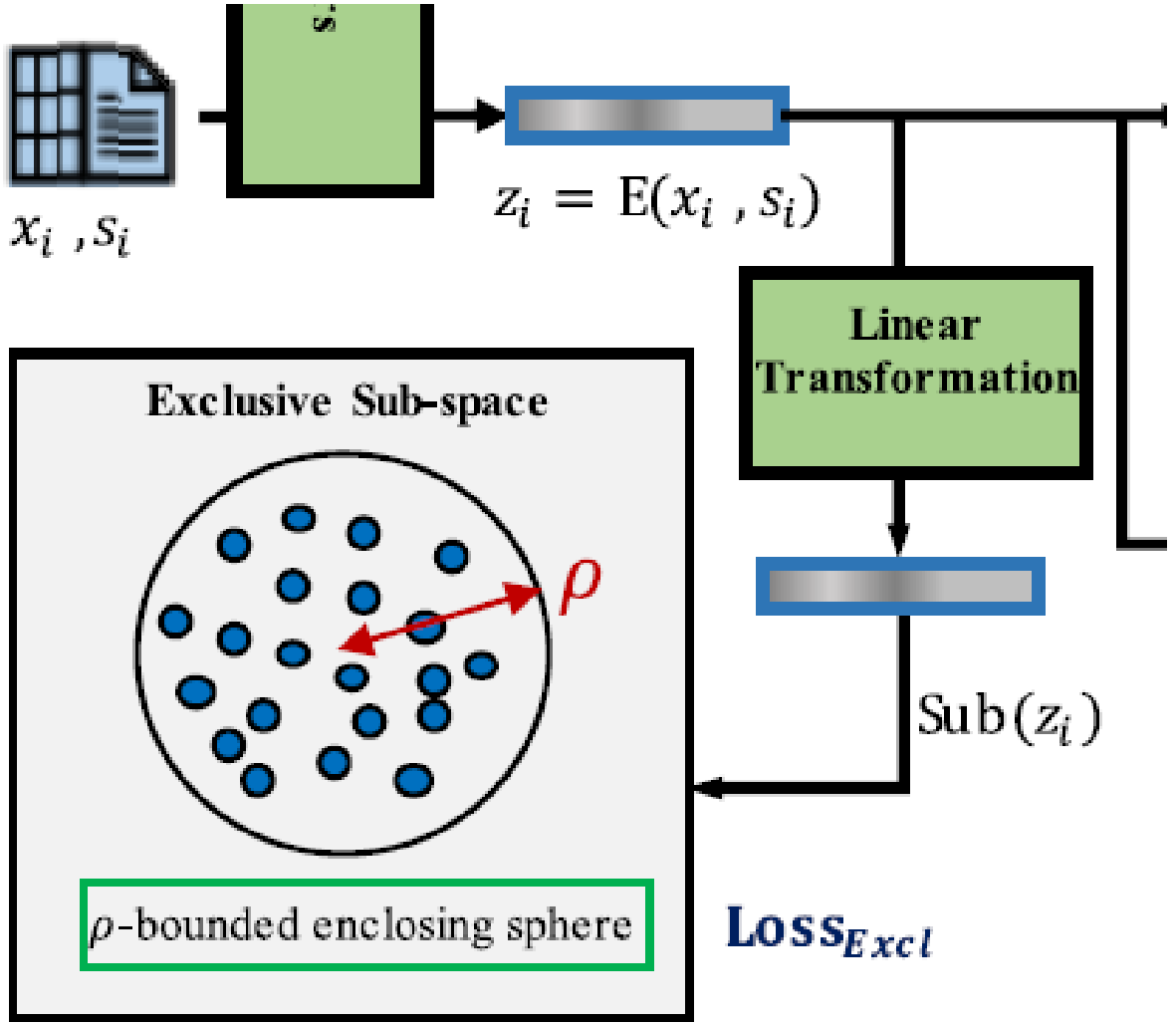
- Öncelikle, bilinen kötü amaçlı yazılım sınıflarının bir minimal kapsayan küre içine sınırlandırılabileceği varsayımı yapılır. Bu küre içinde yer almayan herhangi bir özellik, yeni bir bilinmeyen aileye ait bir örnek olarak kabul edilir.



HATA FONKSİYONU (İkinci Ekleme)

- Öncelikle, bilinen kötü amaçlı yazılım ailelerinin minimal bir kapsayan küre içine yerleştirilmesi ve bu kürenin yarıçapının maksimize edilmesi gerekir. Bu, kürenin içindeki bilinen kötü amaçlı yazılım örneklerinin sayısını artırır.

- Ardından, kürenin yarıçapının minimize edilmesi, böylece bilinen ve bilinmeyen aileler arasındaki özellik sınırının keşfedilmesi.



HATA FONKSİYONU (İkinci Ekleme)

- Elimizde bir kötü amaçlı yazılım (malware) tespit etmek için bir model olsun. Bu model, belirli özelliklerin (dosya boyutu, dosya içeriği, kullanılan işlemler, vb.) malware olup olmadığını belirlemek için kullanılır.

- Öncelikle, modelin öğrenmesi gereken şey, bu özelliklerden yola çıkarak bir dosyanın malware olup olmadığını doğru bir şekilde tahmin etmektir.

- Modelin özelliklerini temsil eden bir uzay oluştururuz. Bu uzay, modelin gözlemlediği ve öğrendiği tüm malware özelliklerini içerir.

HATA FONKSİYONU (İkinci Ekleme)

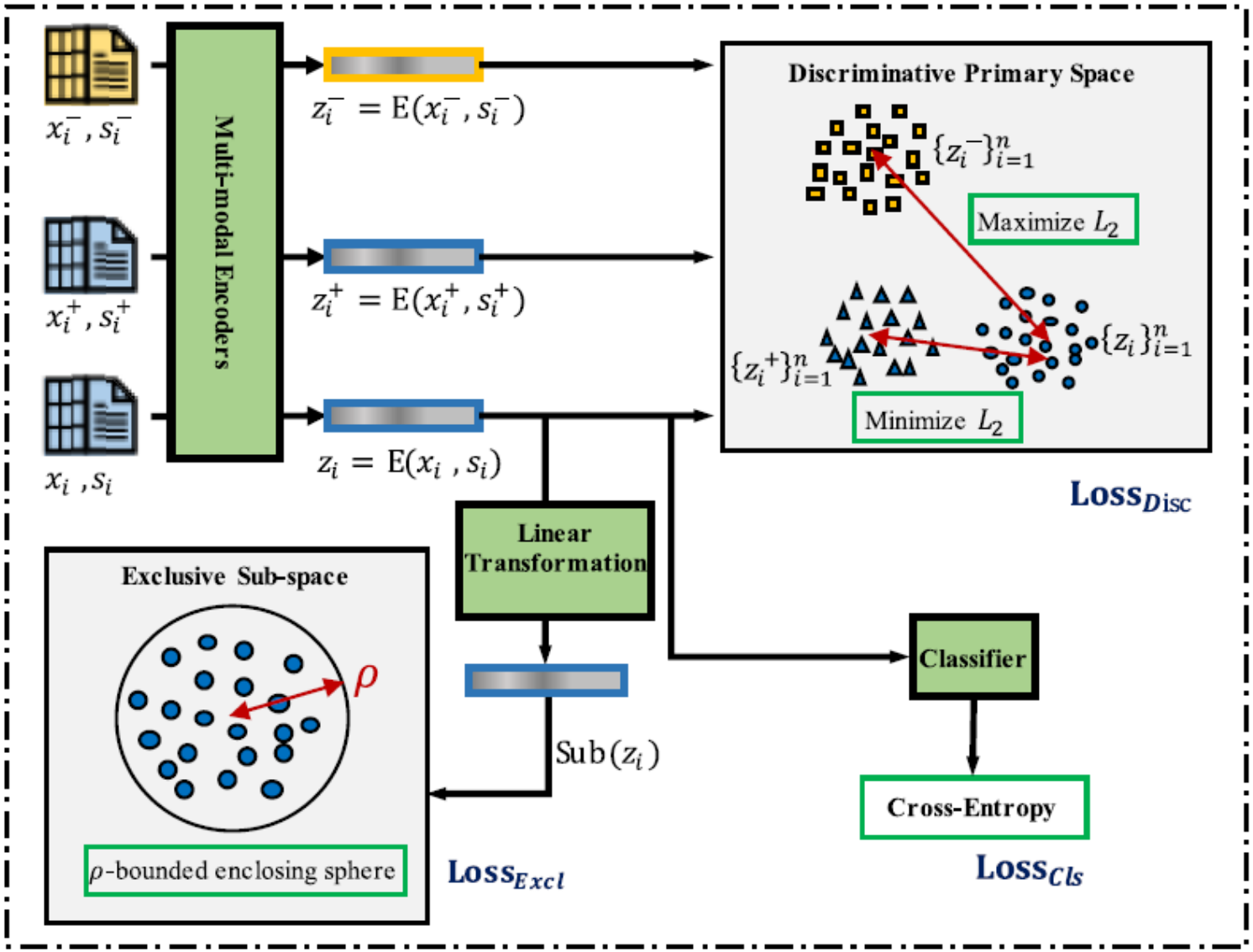
- Elimizdeki malware örneklerinden bazılarını kullanarak bir minimal kapsayan küre tanımlarız. Bu küre, modelin öğrendiği özelliklerin bir kısmını içeren bir bölgedir.
- Bir yeni dosya geldiğinde dosyaya ait özellikler, kürenin içindeyse, yani modelin öğrendiği özelliklere benziyorsa, model bu dosyayı malware olarak tanıyabilir. Ancak, eğer özellikler kürenin dışında ise, yani modelin öğrenmediği bir özellik alanına aitse, model bu dosyayı bilinmeyen bir durum olarak tanımlar.
- Bu şekilde, minimal kapsayan küre, modelin bilinmeyen malware örneklerini tanımasına yardımcı olur. Aynı zamanda, bu küre, modelin öğrendiği özellikleri sınırlar, böylece model sadece bu özelliklere odaklanır ve bilinmeyenleri tespit etme yeteneğini geliştirir.

HATA FONKSİYONU (İkinci Ekleme)

$$\mathrm{Loss}_{\mathrm{Excl}}(\Theta_{\mathrm{Sub}}) = \sum_{i=1}^{n} \max\{\| \mathrm{Sub}(z_i)\|_2 - \rho, 0\} + \rho - \lambda \|\Theta_{\mathrm{Sub}}\|_2.$$

- Burada $Sub(z_i)$, z_i ögesini bir alt uzaya göndermek için yapılan lineer dönüşüm işlemini,
- ρ kürenin yarıçapını temsil eder.
- Model, bilinen malware sınıflarının özelliklerini, kürenin yarıçapının içine girecek minimum şekilde öğrenmeye çalışacak.

HATA FONKSİYONU (Birinci ve İkinci Ekleme)



HATA FONKSİYONU (Final)

$$\begin{aligned}\mathrm{Loss}(\Theta_E, \Theta_C, \Theta_{Sub}) = \alpha \cdot \mathrm{Loss}_{Cls}(\Theta_E, \Theta_C) \\ + \beta \cdot \mathrm{Loss}_{Disk}(\Theta_E) \\ + (1 - \alpha - \beta) \cdot \mathrm{Loss}_{Excl}(\Theta_{Sub})\end{aligned}$$

BİLİNMEYEN YAZILIMLARIN TESPİTİ

- İlk adım, verilen bir kötü amaçlı yazılım örneğinin bilinen ailelere mi yoksa bilinmeyen ailelere mi ait olduğunu tespit etmektir. Model, öncelikle örneğin özelliklerini çıkarır ve bu özellikleri kullanarak tahmini bir aile belirler. Bilinen bir aileye aitse, model bunu doğrudan tahmin eder. Ancak, bilinmeyen bir aileye ait olup olmadığını belirlemek için uzaklık tabanlı bir mekanizma kullanılır.

BİLİNMEYEN YAZILIMLARIN TESPİTİ

- Bu adımda, bilinen sınıfların özellik merkezlerini hesaplamak için eğitim setindeki kötü amaçlı yazılım örnekleri kullanılır. Her sınıf için, örneklerin gömülü özelliklerinin ortalamasını alarak bir özellik merkezi elde edilir (Modelin çıktısı).

$$c_k = \frac{1}{N_k} \cdot \sum_{\mathcal{D}_{\text{tr}}, l = f_k} \mathrm{E}(x, s)$$

BİLİNMEYEN YAZILIMLARIN TESPİTİ

- Daha sonra, bir test örneği verildiğinde, model bu örneğin tahmini ailesini belirler. Ardından, tahmin edilen ailenin özellik merkezi ile test örneği arasındaki uzaklık hesaplanır. Eğer bu uzaklık bir belirlenen eşik değerinden küçükse, model test örneğini bilinen bir aileye atar. Aksi halde, örnek bilinmeyen bir aileye ait olarak kabul edilir.

$$\mathcal{DET}(x) = \begin{cases} \text{Known}, & \| \mathrm{E}(x, s) - c_{\max\{\mathrm{C}(\mathrm{E}(x,s))\}} \|_2 \leq \delta \\ \text{Unknown}, & \| \mathrm{E}(x, s) - c_{\max\{\mathrm{C}(\mathrm{E}(x,s))\}} \|_2 > \delta \end{cases}$$

$$\delta = \max_{\mathcal{D}_{\mathrm{tr}}} \| \mathrm{E}(x, s) - c_l \|_2.$$

SONUÇLAR

TABLE II

RESULTS ON MAILING DATASET

Method	Category	Known		Unknown	Classification	Detection
		Train #	Test #	Test #	Cls_{Acc} (%)	Det_{Acc} (%)
Nataraj *et al.* [50]	TM	8,394	945	-	97.18*	-
Burnaev *et al.* [62]	TM	6,400	1,617	1,322	94.02±0.18	52.61±0.16
Kalash *et al.* [25]	TM	8,394	945	-	93.23*	-
Yajamanam *et al.* [63]	TM	8394	945	-	97.00*	-
GIST+SVM [64]	TM	8,394	945	-	92.20*	-
GLCM+SVM [64]	TM	8,394	945	-	93.20*	-
EVM [39]	TM	6,400	1,617	1,322	98.28±0.15	81.50±0.22
SoftMax-DNNs	DNNs	6,400	1,617	1,322	97.85±0.24	73.31±0.15
tGAN [65]	DNNs	8,394	945	-	96.82*	-
VGG-VeryDeep-19 [66]	DNNs	5,603	1,868	-	97.32*	-
Cui *et al* [64]	DNNs	8,394	945	-	97.60*	-
OpenMax [14]	DNNs	6,400	1,617	1,322	98.62±0.10	84.21±0.13
IDA+DRBA [64]	DNNs	8,394	945	-	94.50*	-
NSGA-II [67]	DNNs	8,394	945	-	97.60*	-
MCNN [25]	DNNs	8,394	945	-	98.52*	-
tDCGAN [68]	DNNs	8,394	945	-	97.66*	-
IMCFN [10]	DNNs	6,537	2,802	-	98.82*	-
Guo *et al.* [20]	DNNs	6,400	1,617	1,322	98.80±0.22	88.79±0.10
OVRNs [34]	DNNs	6,400	1,617	1,322	96.32±0.23	84.51±0.12
MOCCA [69]	DNNs	6,400	1,617	1,322	97.38±0.18	85.31±0.09
FCPN [70]	DNNs	6,400	1,617	1,322	97.44±0.25	86.91±0.17
MDENet (ours)	DNNs	6,400	1,617	1,322	**99.27±0.07**	**90.65±0.12**

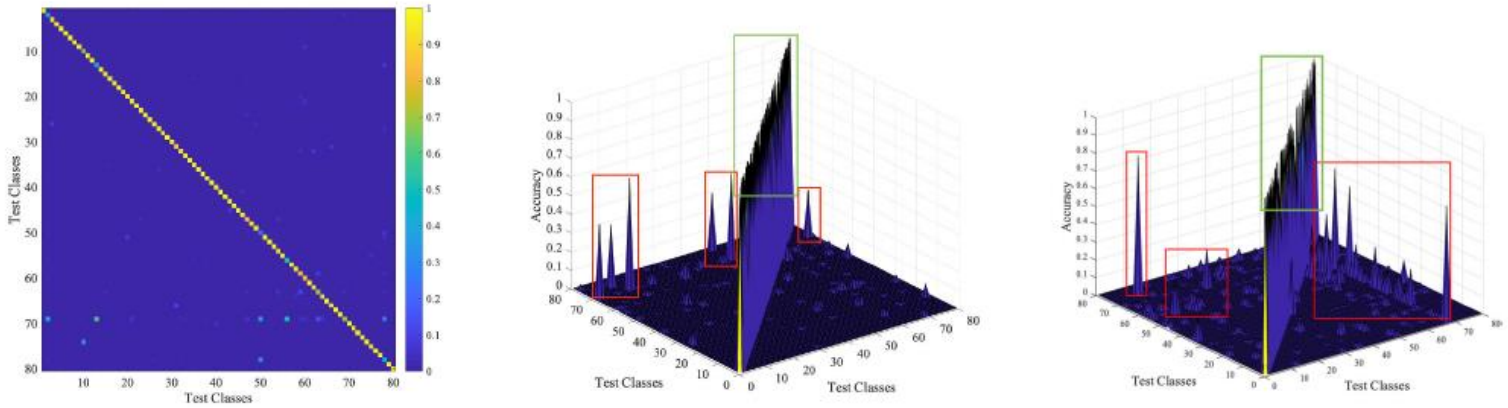
SONUÇLAR

TABLE III

RESULTS ON MAL-100+

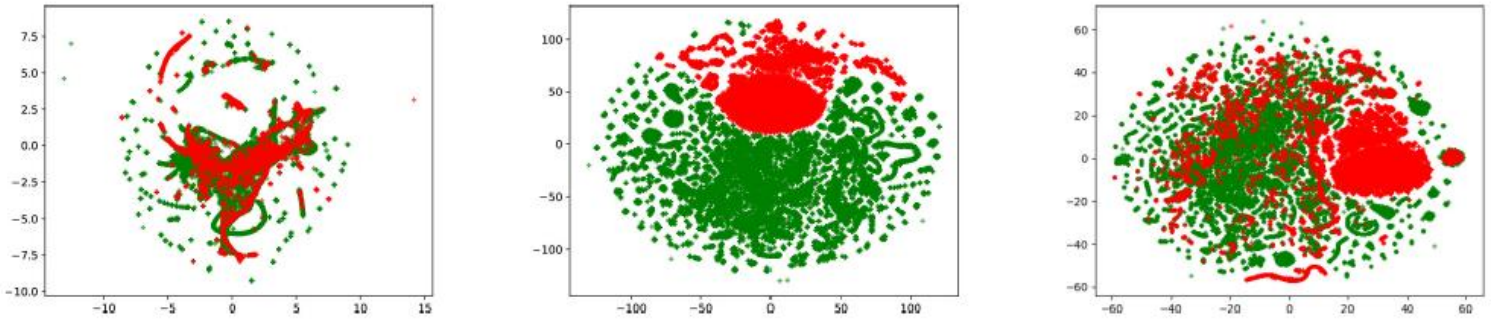
Method	Category	Known		Unknown	Classification	Detection
		Train #	Test #	Test #	Cls_{Acc} (%)	Det_{Acc} (%)
Burnaev *et al.* [62]	TM	31,523	7,823	17,135	32.24±0.35	52.32±0.29
EVM [39]	TM	31,523	7,823	17,135	89.87±0.28	68.48±0.24
SoftMax-DNNs	DNNs	31,523	7,823	17,135	76.03±0.51	62.81±0.38
OpenMax [14]	DNNs	31,523	7,823	17,135	85.21±0.19	70.47±0.10
tGAN [65]	DNNs	31,523	7,823	17,135	76.91±0.35	65.03±0.24
tDCGAN [68]	DNNs	31,523	7,823	17,135	81.21±0.27	66.86±0.21
Guo *et al.* [20]	DNNs	31,523	7,823	17,135	90.94±0.25	86.07±0.19
OVRNs [34]	DNNs	31,523	7,823	17,135	87.25±0.37	82.05±0.29
MOCCA [69]	DNNs	31,523	7,823	17,135	90.11±0.22	84.12±0.17
FCPN [70]	DNNs	31,523	7,823	17,135	88.85±0.24	84.00±0.21
MDENet (ours)	DNNs	31,523	7,823	17,135	**94.14±0.17**	**90.23±0.19**

SONUÇLAR



(a) (b) (c)

Fig. 6. Per-family classification performance of the proposed MDENet on MAL-100$^+$ (a) classification confusion matrix and (b) landscape of the confusion matrix. Moreover, we also demonstrate the landscape of the current state-of-the-art competitor, i.e., Guo et al. [20], in (c) (better viewed in color). (a) Classification confusion matrix (ours). (b) Landscape (ours). (c) Landscape (Guo et al. [20]).



(a) (b) (c)

Fig. 7. t-SNE visualization of malware samples in MAL-100$^+$ (a) original samples without processing, (b) mapped samples by the proposed method, and (c) mapped samples by Guo et al. [20]. Green and red "+" denote known and unknown malware families, respectively (better viewed in color). (a) Original malware samples. (b) Mapped malware samples (Ours). (c) Mapped malware samples (Guo et al. [20]).

ANKARA ÜNİVERSİTESİ
SİBER GÜVENLİK MESLEK YÜKSEKOKULU

Çift Modüllü Ağlar ile Kötü Yazılım (Malware) Açık-küme Tanıma

Multimodal Dual-Embedding Networks for Malware Open-Set Recognition