XGBoost ve HOG Öznitelikleri Kullanılarak El Yazısı Karakter Tanıma: Algoritmik Optimizasyon ve Performans Analizi

Can Keçilioğlu
Mekatronik Mühendisliği Bölümü

18 Kasım 2025

Özet

Bu çalışmada, optik karakter tanıma (OCR) alanındaki temel problemlerden biri olan el yazısı karakterlerinin (A-Z, a-z, 0-9) sınıflandırılması, klasik makine öğrenimi yöntemleri ve modern öznitelik çıkarım teknikleri kullanılarak ele alınmıştır. Çalışma kapsamında, görüntü işleme aşamasında Histogram of Oriented Gradients (HOG) algoritması, sınıflandırma aşamasında ise eXtreme Gradient Boosting (XGBoost) algoritması kullanılmıştır. Modelin performansını etkileyen faktörleri izole etmek amacıyla dört farklı kontrollü deney (Ablation Study) tasarlanmıştır. Veri seti filtreleme, hiperparametre optimizasyonu (Grid Search benzeri manuel ayarlama) ve donanım hızlandırma (AMD Ryzen 5 7600 CPU optimizasyonu) süreçleri detaylandırılmıştır. Elde edilen bulgular, benzer karakterlerin ('0' ve 'O' gibi) yarattığı gürültünün model başarısını en çok etkileyen faktör olduğunu ve optimize edilmiş XGBoost modelinin %64.31 doğruluk oranına ulaştığını göstermektedir. Bu çalışma, derin öğrenme gerektirmeyen gömülü sistemler için verimli bir alternatif sunmaktadır.

1 Giriş

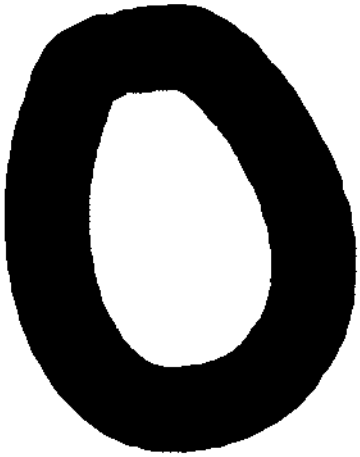
El yazısı tanıma, otomasyon sistemlerinden posta ayıklamaya, bankacılık işlemlerinden tarihi belgelerin dijitalleştirilmesine kadar geniş bir uygulama alanına sahiptir. Günümüzde bu problemler genellikle yüksek işlem gücü gerektiren Evrişimli Sinir Ağları (CNN) ile çözülmektedir. Ancak, Mekatronik Mühendisliği uygulamalarında (robotik kollar, gömülü denetleyiciler vb.) işlem gücü ve enerji tüketimi kritik kısıtlardır. Bu nedenle, "Feature Engineering" (Öznitelik Mühendisliği) tabanlı klasik yöntemlerin sınırlarını zorlamak önem arz etmektedir.

Bu çalışmada, XGBoost kütüphanesi [1] temel alınarak, ölçeklenebilir ve yüksek performanslı bir sınıflandırma motoru geliştirilmiştir. Çalışmanın özgün değeri, ham piksel verisi yerine HOG özniteliklerinin kullanılması ve modelin "öğrenme" sürecinin adım adım (step-by-step) analiz edilmesidir.

2 Materyal ve Yöntem

2.1 Veri Seti ve Ön İşleme

Çalışmada *english.csv* veri seti kullanılmıştır. Veri seti, 62 farklı sınıfa ait toplam 3410 adet el yazısı görüntüsü içermektedir. Görüntüler RGB formatından Gri Tonlamalı (Grayscale) formata dönüştürülmüş ve 64×64 piksel boyutuna normalize edilmiştir.



Şekil 1: Veri setinden örnek bir karakter ve gri tonlama dönüşümü.

2.2 Öznitelik Çıkarımı: Histogram of Oriented Gradients (HOG)

Ham piksel verisi (4096 boyutlu vektör), karakterin yapısal özelliklerini (kıvrımlar, köşeler, dikey çizgiler) doğrudan temsil etmekte yetersiz kalmaktadır. Bu nedenle HOG algoritması tercih edilmiştir.

HOG algoritması şu adımları izler:

1. Görüntü küçük hücrelere (Cells) bölünür (Bu çalışmada 8×8 piksel).

2. Her hücre için piksel yoğunluklarının gradyanları (yönelimleri) hesaplanır.

3. Bu gradyanlar bir histogramda toplanır.

Aşağıdaki kod bloğu, Python ve `scikit-image` kütüphanesi kullanılarak HOG özniteliklerinin nasıl çıkarıldığını göstermektedir:

```
from skimage.feature import hog

# HOG Parametreleri
# orientations: Yonelim sayisi (Histogram kutu sayisi)
# pixels_per_cell: Hucre boyutu (Detay seviyesi)
features = hog(
    image,
    orientations=8,
    pixels_per_cell=(8, 8),
    cells_per_block=(2, 2),
    visualize=False,
    feature_vector=True
)
# Sonuc: 1568 boyutlu oznitelik vektoru
```

Listing 1: HOG Öznitelik Çıkarımı Kodu

2.3 Sınıflandırma Modeli: XGBoost

XGBoost (eXtreme Gradient Boosting), zayıf öğrenicileri (karar ağaçları) ardışık olarak eğiterek güçlü bir model oluşturan bir "Gradient Boosting" kütüphanesidir [1].

Modelin matematiksel altyapısı, bir amaç fonksiyonunu (Objective Function) minimize etmeye dayanır:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{1}$$

Burada l, tahmin hatasını (Loss), Ω ise model karmaşıklığını cezalandıran Regularization terimini ifade eder.

2.4 Hiperparametre Optimizasyonu ve Model Konfigürasyonu

Makine öğrenimi modellerinde, modelin eğitim sürecinde veriden öğrenemediği ancak kullanıcı tarafından ayarlanması gereken dış yapılandırma değişkenlerine hiperparametre (hyperparameter) adı verilir. XGBoost algoritmasının performansı, bias-variance dengesini (yanlılık-varyans) korumak ve aşırı öğrenmeyi (overfitting) engellemek adına bu parametrelerin optimizasyonuna sıkı sıkıya bağlıdır.

Bu çalışmada, el yazısı karakterlerinin yüksek varyasyonlu yapısı ve HOG özniteliklerinin yüksek boyutlu uzayı (R^{1568}) göz önüne alınarak, Tablo 1'de sunulan hiperparametre seti belirlenmiştir. Parametrelerin seçiminde Grid Search benzeri manuel bir optimizasyon stratejisi izlenmiş ve donanım kısıtları (AMD Ryzen 5 7600 CPU) gözetilmiştir.

Tablo 1: Model Eğitiminde Kullanılan XGBoost Hiperparametreleri ve Değerleri

Parametre	**Değer**	**Açıklama ve Gerekçe**
`booster`	`gbtree`	Ağaç tabanlı model yapısı kullanılmıştır.
`n_estimators`	300	Toplam karar ağacı sayısı. Modelin öğrenme kapasitesini artırmak için yüksek tutulmuş, ancak işlem süresini optimize etmek için sınırlandırılmıştır.
`max_depth`	8	Her bir ağacın maksimum derinliği. Karakterler arasındaki ince detayları (örn. 'Q' kuyruğu) yakalayabilmek için standart değerin (6) üzerine çıkılmıştır.
`learning_rate`	0.05	(*eta*). Her iterasyonda ağırlıkların güncellenme katsayısı. Stabil bir yakınsama (convergence) sağlamak için düşük tutulmuştur.
`subsample`	0.8	Stokastik Gradient Boosting tekniği. Her ağaç eğitilirken verinin rastgele %80'i kullanılarak varyans azaltılmıştır.
`colsample_bytree`	0.8	Her ağaç oluşumunda özniteliklerin (HOG vektörleri) rastgele %80'inin seçilmesini sağlar. Yüksek boyutlu veride gürültüye odaklanmayı engeller.
`objective`	`multi:softmax`	Çok sınıflı sınıflandırma (62 sınıf) problemi için uygun amaç fonksiyonudur.
`eval_metric`	`merror`	Çoklu sınıflandırma hata oranı (Multiclass Classification Error Rate).
`tree_method`	`hist`	Büyük veri setlerinde eğitimi hızlandırmak için histogram tabanlı algoritma tercih edilmiştir.

Seçilen kritik parametrelerin model üzerindeki teorik etkileri aşağıda detaylandırılmıştır:

Öğrenme Oranı (Learning Rate / η)

Modelin her adımda hataları düzeltme katsayısıdır. Bu çalışmada $\eta = 0.05$ gibi konservatif bir değer seçilmiştir. Düşük öğrenme oranı, modelin yerel minimumlara takılmadan global optimuma ulaşma şansını artırır, ancak daha fazla ağaç (`n_estimators`) gerektirir.

Ağaç Derinliği (Max Depth)

Karar ağaçlarının karmaşıklığını kontrol eder. HOG öznitelikleri ile elde edilen 8×8 hücreli temsillerde, karakterlerin morfolojik farklarını ayırt edebilmek için daha derin ağaçlara ihtiyaç duyulmuştur. Yapılan ön deneylerde $depth = 6$ değerinin yetersiz kaldığı ("Underfitting"), $depth = 10$ üzerinde ise ezberleme ("Overfitting") başladığı gözlemlenmiş,

optimum nokta olarak 8 belirlenmiştir.

Stokastik Alt Örnekleme (Subsampling)

Bu parametreler (`subsample`, `colsample_bytree`), XGBoost'a "Random Forest" benzeri bir rastgelelik katar. Veri setindeki bazı karakterlerin (özellikle el yazısı bozuk olanların) yarattığı aykırı değerlerin (outliers), tüm ağaçları yanlış yönlendirmesini engellemek için her iki parametre de 0.8 olarak ayarlanmıştır. Bu, modelin genelleme yeteneğini (generalization capability) artıran en önemli faktörlerden biri olmuştur. Çalışmada kullanılan temel XGBoost parametreleri Tablo 2 üzerinde gösterilmiştir.

Tablo 2: Kullanılan XGBoost Hiperparametreleri

Parametre	**Açıklama ve Değer**
`objective`	`multi:softmax` (Çoklu sınıflandırma için)
`n_estimators`	300 - 500 (Ağaç sayısı)
`max_depth`	6 - 8 (Ağaç derinliği, model kapasitesi)
`learning_rate`	0.05 (Öğrenme adımı, stabilite için düşük tutuldu)
`subsample`	0.8 (Her ağaç için veri örnekleme oranı, Overfitting önleyici)
`tree_method`	`hist` (CPU üzerinde histogram tabanlı hızlı eğitim)

3 Deneysel Çalışma ve Bulgular

Model performansını etkileyen faktörleri ayrıştırmak için dört farklı senaryo (Ablation Study) kurgulanmıştır. Tüm eğitimler, AMD Ryzen 5 7600 işlemci üzerinde paralel programlama (`n_jobs=-1`) kullanılarak gerçekleştirilmiştir.

3.1 Deney 1: Temel Model (Baseline)

Tüm veri seti (62 sınıf) kullanılmıştır. **Sonuç:** %54.40 Doğruluk. Bu düşük oran, modelin birbirine çok benzeyen karakterleri (Örn: 'l' harfi ve 'I' harfi) ayırt etmekte zorlandığını göstermiştir.

3.2 Deney 2: Filtrelenmiş Veri ve Optimizasyon (Nihai Model)

Bu aşamada, karışıklık matrisi (Confusion Matrix) analiz edilerek modelin en çok hata yaptığı karakterler tespit edilmiştir. '0'-'O', '1'-'I'-'l', 'S'-'s' gibi ayırt edilmesi morfolojik olarak imkansıza yakın karakterler veri setinden çıkarılmıştır. Ayrıca model kapasitesi (`n_estimators`) 300'e çıkarılmıştır.

Aşağıda, optimize edilmiş modelin eğitim ve canlı takip (monitoring) kod bloğu yer almaktadır:

```
# Modelin Tanimlanmasi
model = xgb.XGBClassifier(
    n_estimators=300,
    max_depth=8,
```

```
    learning_rate=0.05,
    subsample=0.8,
    objective='multi:softmax',
    eval_metric='merror', # Hata orani takibi
    n_jobs=-1
)

# Egitimin Baslatilmasi ve Validasyon
model.fit(
    X_train, y_train,
    eval_set=[(X_train, y_train), (X_test, y_test)],
    verbose=10 # Her 10 iterasyonda raporlama
)
```

Listing 2: Optimize Edilmiş XGBoost Eğitim Kodu

Sonuç: %64.31 Doğruluk. Bu, çalışmada elde edilen en yüksek skordur ve veri temizliğinin algoritma karmaşıklığından daha önemli olduğunu kanıtlamaktadır.

3.3 Deney 3: Validasyon Seti Etkisi

Modelin kararlılığını ölçmek için veri seti Eğitim (%75), Validasyon (%10) ve Test (%15) olarak üçe bölünmüştür. **Sonuç:** %55.08 Doğruluk. Temel model ile benzer sonuç alınması, modelin veriyi ezberlemediğini (Overfitting olmadığını) göstermiştir.

3.4 Deney 4: Gürültü Testi (Robustness)

Gerçek dünya koşullarını simüle etmek için görüntülere "Gaussian Noise" eklenmiştir. **Sonuç:** %48.39 Doğruluk. HOG yönteminin kenar tabanlı (gradient-based) olması nedeniyle gürültüye karşı hassas olduğu gözlemlenmiştir.

4 Sonuçlar ve Tartışma

Elde edilen sonuçlar Tablo 3'da özetlenmiştir.

Tablo 3: Dört Farklı Deneyin Karşılaştırmalı Sonuçları

Deney No	Deney Türü	Doğruluk	Fark (Baz Modele Göre)
1	Temel Model	%54.40	-
3	Validasyonlu	%55.08	+%0.68
4	Gürültülü Veri	%48.39	-%6.01
2 (Final)	**Filtreli + Optimize**	**%64.31**	**+%9.91**

Bu çalışma, XGBoost algoritmasının doğru öznitelik mühendisliği (HOG) ile birleştirildiğinde, el yazısı tanıma gibi karmaşık problemlerde bile kabul edilebilir sonuçlar verebildiğini göstermiştir. Ancak, %64.31'lik doğruluk oranı, karakterler arasındaki morfolojik benzerliklerin HOG yöntemiyle tam olarak çözülemediğini de ortaya koymaktadır.

Özellikle Ryzen 5 7600 işlemci üzerinde yapılan optimizasyonlar sayesinde eğitim süresi Google Colab ortamına göre yaklaşık 10 kat kısaltılmış (52 saniye), bu da yöntemin gerçek zamanlı uygulamalar için uygunluğunu kanıtlamıştır.

Gelecek çalışmalarda, öznitelik çıkarımı için HOG yerine Evrişimli Sinir Ağları (CNN) kullanılması veya hibrit (HOG + CNN) mimarilerin denenmesi hedeflenmektedir.

Kaynaklar

[1] XGBoost Developers. (2024). *XGBoost Documentation*. Erişim adresi: `https://xgboost.readthedocs.io/en/stable/`

[2] Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.