T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CNN MİMARİLERİNİ KULLANARAK KENDİ KENDİNE ÖĞRENEN SÖZDE ETİKETLEME ALGORİTMASI İÇİN YENİ BİR KOLEKTİF YAKLAŞIM

Mehmet AKDİŞ

YÜKSEK LİSANS TEZİ
Bilgisayar Mühendisliği Anabilim Dalı
Bilgisayar Mühendisliği Programı

Danışman
Prof. Dr. Banu DİRİ

Eş-Danışman
Öğr. Gör. Dr. Ahmet ELBİR

Mayıs, 2025

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

CNN MİMARİLERİNİ KULLANARAK KENDİ KENDİNE ÖĞRENEN SÖZDE ETİKETLEME ALGORİTMASI İÇİN YENİ BİR KOLEKTİF YAKLAŞIM

Mehmet AKDİŞ tarafından hazırlanan tez çalışması 22.05.2025 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Bilgisayar Mühendisliği Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. Banu DİRİ
Yıldız Teknik Üniversitesi
Danışman

Öğr. Gör. Dr. Ahmet ELBİR
Yıldız Teknik Üniversitesi
Eş-Danışman

Jüri Üyeleri

Prof. Dr. Banu DİRİ, Danışman
Yıldız Teknik Üniversitesi _____

Doç. Dr. Ali Can KARACA, Üye
Yıldız Teknik Üniversitesi _____

Dr. Öğr. Üyesi Nihal ALTUNTAŞ, Üye
İstanbul Gelişim Üniversitesi _____

Danışmanım Prof. Dr. Banu DİRİ sorumluluğunda tarafımca hazırlanan CNN Mimarilerini Kullanarak Kendi Kendine Öğrenen Sözde Etiketleme Algoritması için Yeni Bir Kolektif Yaklaşım başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Mehmet AKDİŞ

İmza

Her zaman yanımda olan sevgili aileme

TEŞEKKÜR

Bu çalışmanın gerçekleştirilmesinde, yardımcı olan değerli tez danışmanım Prof. Dr. Banu DİRİ ve eş danışmanım Öğr. Gör. Dr. Ahmet ELBİR'e, derslerini hayranlıkla dinlediğim ve kendisinden çok şey öğrendiğim Prof. Dr. Mehmet Fatih AMASYALI'ya ve 2210-A Genel Yurt İçi Yüksek Lisans Burs Programı ile burs desteği sağlayan TÜBİTAK Bilim İnsanı Destek Programları Başkanlığı'na teşekkürlerimi sunarım.

Mehmet AKDİŞ

İÇİNDEKİLER

ŞEKİL LİSTESİ vii

TABLO LİSTESİ ix

ÖZET xi

ABSTRACT xiii

1 GİRİŞ 1

1.1 Tezin Amacı . . . 1
1.2 Literatür Araştırması . . . 3
1.3 Hipotez . . . 5

2 ÖNERİLEN YÖNTEM 6

2.1 Kendi Kendine Öğrenme . . . 6
2.2 Yöntemin Tanımı . . . 6
2.2.1 ϕ_E Olasılıklarının Hesaplanması . . . 7
2.2.2 ϕ Olasılıklarının Hesaplanması . . . 7
2.3 Her İterasyondaki Ekstra Çıktılar . . . 9
2.4 Bağımsızlıklar . . . 9
2.5 Neden Olasılıklar Kaydediliyor? . . . 9
2.6 Değişkenler ve Algoritmalar . . . 10
2.7 Kısıtların Tanımlanması . . . 16

3 DENEYSEL SONUÇLAR 17

3.1 Veri Setlerinin Tanıtımı . . . 19
3.1.1 CIFAR10 . . . 19
3.1.2 Intel . . . 20
3.1.3 Gtzan . . . 20
3.2 Sonuçlar . . . 22
3.2.1 Genel Sonuçlar . . . 23
3.2.2 Var Olan Yöntemin Sonuçları . . . 24
3.2.3 α Değerinin Etkisi . . . 28

3.2.4 Adaptif α Yaklaşımı 35
3.2.5 Farklı α Değerlerinin Karşılaştırılması 38
3.2.6 Farklı Mimari Bağımsızlığı 40
3.2.7 Intel Veri Kümesi Deneysel Sonuçlar 42
3.2.8 Gtzan Veri Kümesi Deneysel Sonuçlar 46

4 KSIITLARIN İNCELENMESİ 49
4.1 Olasılık Şartında Neden Azalma Yapılıyor ? 49
4.2 Akış Neden Yüzdelik Olarak Sonlandırılıyor ? 49
4.3 Önerilen Yöntemde Neden Son Yedi Olasılık Kullanılıyor ? 50
4.4 Süre Faktörünün İncelenmesi 50
4.4.1 Birinci Durum 50
4.4.2 İkinci Durum 52

5 DENEYSEL BULGULARIN TARTIŞILMASI 54

6 SONUÇ 56

KAYNAKÇA 57

TEZDEN ÜRETİLMİŞ YAYINLAR 60

ŞEKİL LİSTESİ

Şekil 2.1 Önerilen yaklaşımın akış şeması. 8
Şekil 3.1 CIFAR10 Veri Seti 19
Şekil 3.2 Intel Veri Seti 20
Şekil 3.3 Gtzan Veri Setinin Ayarlanması 22
Şekil 3.4 CİFAR10 sonuçları. Üst satır EfficientNetBO, alt satır DenseNet121 mimarisine aittir. 26
Şekil 3.5 Intel sonuçları. Üst satır EfficientNetBO, alt satır DenseNet121 mimarilerine aittir. 26
Şekil 3.6 Gtzan sonuçları. 27
Şekil 3.7 α=0,1 İterasyonların Sonuçları 28
Şekil 3.8 α=0,2 İterasyonların Sonuçları 29
Şekil 3.9 α=0,3 İterasyonların Sonuçları 30
Şekil 3.10 α=0,4 İterasyonların Sonuçları 31
Şekil 3.11 α=0,5 İterasyonların Sonuçları 31
Şekil 3.12 α=0,6 İterasyonların Sonuçları 32
Şekil 3.13 α=0,7 İterasyonların Sonuçları 33
Şekil 3.14 α=0,8 İterasyonların Sonuçları 34
Şekil 3.15 α=0,9 İterasyonların Sonuçları 35
Şekil 3.16 Adaptif α İterasyonların Sonuçları 37
Şekil 3.17 Doğrulama Verisindeki Ortalama Başarımlar 39
Şekil 3.18 Test Verisindeki Ortalama Başarımlar 39
Şekil 3.19 Ortalama Nihai Süre ve Başarımların Değişimleri 40
Şekil 3.20 Adaptif α İterasyonların Sonuçları 41
Şekil 3.21 Test ve Doğrulama Verilerindeki Ortalama Başarımlar 42
Şekil 3.22 EfficientNetB0 İterasyonların Sonuçları 44
Şekil 3.23 DenseNet121 İterasyonların Sonuçları 44
Şekil 3.24 Var olan yöntem ile karşılaştırmalı iterasyonların ortalama sonuçları. Üst satır EfficientNetB0, alt satır DenseNet121 mimarilerine aittir. 45
Şekil 3.25 ResNet34 İterasyonların Sonuçları 47
Şekil 3.26 Var olan yöntem ile karşılaştırmalı iterasyonların ortalama sonuçları. 47

Şekil 4.1 Birinci Durum İterasyonların Sonuçları 51
Şekil 4.2 İkinci Durum İterasyonların Sonuçları 52
Şekil 4.3 Birinci ve İkinci Durumlarındaki İterasyonların Ortalama Sonuçları 53

TABLO LİSTESİ

Tablo 2.1 Global Değişkenlerin Tanımları 10
Tablo 2.2 Fonksiyonlar ve Görevleri 11
Tablo 2.3 Kısıtlar 16
Tablo 3.1 CIFAR10 Genel Sonuçları 23
Tablo 3.2 Gtzan Genel Sonuçları 23
Tablo 3.3 Intel Genel Sonuçları 24
Tablo 3.4 CIFAR10 Var Olan Yöntem Sonuçları 25
Tablo 3.5 Intel Var Olan Yöntem Sonuçları 25
Tablo 3.6 Gtzan Var Olan Yöntem Sonuçları 26
Tablo 3.7 CIFAR10 Var Olan Yöntemin En İyi İterasyon Sonuçları 27
Tablo 3.8 Intel Var Olan Yöntemin En İyi İterasyon Sonuçları 27
Tablo 3.9 Gtzan Var Olan Yöntemin En İyi İterasyon Sonuçları 27
Tablo 3.10 α=0,1 Nihai Sonuçları 28
Tablo 3.11 α=0,1 En Başarılı Andaki Doğruluk Sonuçları 28
Tablo 3.12 α=0,2 Nihai Sonuçları 29
Tablo 3.13 α=0,2 En Başarılı Andaki Doğruluk Sonuçları 29
Tablo 3.14 α=0,3 Nihai Sonuçları 29
Tablo 3.15 α=0,3 En Başarılı Andaki Doğruluk Sonuçları 30
Tablo 3.16 α=0,4 Nihai Sonuçları 30
Tablo 3.17 α=0,4 En Başarılı Andaki Doğruluk Sonuçları 30
Tablo 3.18 α=0,5 Nihai Sonuçları 31
Tablo 3.19 α=0,5 En Başarılı Andaki Doğruluk Sonuçları 31
Tablo 3.20 α=0,6 Nihai Sonuçları 32
Tablo 3.21 α=0,6 En Başarılı Andaki Doğruluk Sonuçları 32
Tablo 3.22 α=0,7 Nihai Sonuçları 33
Tablo 3.23 α=0,7 En Başarılı Andaki Doğruluk Sonuçları 33
Tablo 3.24 α=0,8 Nihai Sonuçları 33
Tablo 3.25 α=0,8 En Başarılı Andaki Doğruluk Sonuçları 34
Tablo 3.26 α=0,9 Nihai Sonuçları 34
Tablo 3.27 α=0,9 En Başarılı Andaki Doğruluk Sonuçları 34
Tablo 3.28 Adaptif α Nihai Sonuçları 37
Tablo 3.29 Adaptif α En Başarılı Andaki Doğruluk Sonuçları 37

Tablo 3.30 CIFAR-10 Var Olan Yönteme Göre Yüzdelik Başarımları 38
Tablo 3.31 Adaptif α Nihai Sonuçları 41
Tablo 3.32 Adaptif α En Başarılı Andaki Doğruluk Sonuçları 41
Tablo 3.33 Intel Nihai Sonuçları 43
Tablo 3.34 Intel En Başarılı Andaki Doğruluk Sonuçlar 43
Tablo 3.35 Intel Var Olan Yönteme Göre Yüzdelik Başarımları 45
Tablo 3.36 GTZAN Nihai Sonuçları 46
Tablo 3.37 GTZAN En Başarılı Andaki Doğruluk Sonuçlar 46
Tablo 3.38 GTZAN Var Olan Yönteme Göre Yüzdelik Başarımları 47
Tablo 4.1 Birinci Durum Nihai Sonuçları 51
Tablo 4.2 Birinci Durum En Başarılı Andaki Doğruluk Sonuçları 51
Tablo 4.3 İkinci Durum Nihai Sonuçları 52
Tablo 4.4 İkinci Durum En Başarılı Andaki Doğruluk Sonuçları 52

ÖZET

CNN MİMARİLERİNİ KULLANARAK KENDİ KENDİNE ÖĞRENEN SÖZDE ETİKETLEME ALGORİTMASI İÇİN YENİ BİR KOLEKTİF YAKLAŞIM

Mehmet AKDİŞ

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lİsans Tezi

Danışman: Prof. Dr. Banu DİRİ
Eş-Danışman: Öğr. Gör. Dr. Ahmet ELBİR

Yarı gözetimli öğrenme, hem etiketli hem de etiketsiz örneklemleri bir arada kullanarak gözetimli ve gözetimsiz öğrenme yöntemlerini birleştiren bir yaklaşımdır. Bu yaklaşımın temel amacı, etiketsiz örneklemlerden bilgi çıkartarak makine öğrenmesi yönteminin (modelin) tahmin performansını arttırmaktır. Bu kapsamda geliştirilen algoritmalar arasında, yaygın olarak kullanılan yöntemlerden biri de kendi kendine öğrenme (self-learning) algoritmasıdır. Kendi kendine öğrenme algoritması, modelin etiketli örneklemler ile eğitilmesiyle başlar. Eğitim süreci tamamlanan modelin, etiketsiz örneklemler üzerindeki tahmin olasılıkları hesaplanır. Elde edilen bu olasılıklar önceden tanımlanmış olan bir eşik değeri ile kıyaslanır. Eşik değerinden daha yüksek olasılığa sahip örneklemlerin sözde etiketleri (pseudo-labels) üretilir. Sözde etiketleri üretilen bu örneklemler, etiketsiz örneklemler arasından çıkartılarak etiketli örneklemler arasına eklenir. Bir sonraki iterasyonda model, güncellenmiş etiketli örneklemlerle en baştan eğitilir. Bu süreç, etiketsiz örneklem kalmayana kadar iteratif bir şekilde tekrarlanır. Önerilen yöntemde ise kendi kendine öğrenme algoritmasına kolektiflik kazandırılmıştır. Bu kapsamda, sözde etiket üretiminde sadece anlık modelin tahminlerini kullanmak yerine geçmiş iterasyonlardaki modellerin oluşturduğu kolektif tahminde kullanılmıştır. Bu kolektif yaklaşım, Momentumlu Stokastik Gradyan İnişi (SGDM) algoritmasındaki geçmiş gradyanların oluşturduğu hız

(velocity) vektöründen esinlenerek kurulmuştur. Her iterasyon sonunda eğitilen modelin etiketsiz veri üzerindeki olasılık tahminleri depolanır. Belli bir iterasyon anı için hem anlık iterasyondaki modelin olasılık tahminleri hem de depolanmış olasılık tahminlerinin birlikte oluşturduğu kolektif sonuç birleştirilir. Depolanmış olasılıkların kolektif sonucu bulunurken her bir modelin tahmin olasılıkları, doğrulama örneklemleri üzerindeki doğruluk değeri ile ağırlıklandırılmıştır. Böylece daha başarılı tahminlerin etkisi arttırılmışken, başarısız olan tahminlerin etkisi azaltılmıştır. Birleştirme aşamasında ise anlık modelin olasılık tahminleri $(1-0,1), (1-0,2), \ldots, (1-0,9)$ ile geçmiş modellerin oluşturduğu kolektif sonuç $0,1, 0,2, \ldots, 0,9$ tam sayıları ile ağırlıklandırılmıştır. Ayrıca adaptif bir şekilde de ağırlıklandırma da yapılmıştır. Birleştirme sonrası elde edilen nihai olasılıklar ise sözde etiket üretiminde kullanılmıştır. Böylelikle sözde etiketlerin üretiminde hem anlık model hem de geçmiş iterasyonlardaki modeller kullanılmış olur. Önerilen bu algoritma, iki aşamadan meydana gelen kolektif yaklaşımıyla, kendi kendine öğrenen algoritmasına yenilik katmıştır. Önerilen bu yaklaşım, EfficientNetB0, DenseNet121 ve ResNet34 mimarileri ile CIFAR-10, Intel Image Classification ve Gtzan veri kümelerinde test edilmiştir. En başarılı sonuçlar, adaptif ağırlıklandırma ile alınmıştır. Etiketsiz örneklemlerin tamamının etiketlendiği deneysel bulgulara göre önerilen yöntem, aynı süre içinde, kullanılan mimari ve veri setine bağlı olarak ortalama %5,35-%14,82 daha yüksek doğruluğa sahip olmuştur. Bütün süreç içerisindeki en iyi iterasyon anındaki doğruluk değerlerinde ise önerilen yöntem, kullanılan mimari ve veri setine bağlı olarak ortalama %3,58-%8,81 daha yüksek doğruluk vermiştir.

Anahtar Kelimeler: Kendi kendine öğrenme, kolektif öğrenme, sözde etiketleme, yarı gözetimli öğrenme

ABSTRACT

A NEW ENSEMBLE APPROACH FOR SELF-LEARNING PSEUDO-LABELING ALGORITHM WITH USING CNN ARCHITECTURES

Mehmet AKDİŞ

Department of Computer Engineering
Master of Science Thesis

Supervisor: Prof. Dr. Banu DIRI
Co-supervisor: Lect. PhD Ahmet ELBIR

Semi-supervised learning is an approach that combines supervised and unsupervised learning methods by utilizing both labeled and unlabeled samples together. The primary objective of this approach is to enhance the predictive performance of the machine learning method (model) by extracting information from unlabeled samples. Among the widely used algorithms developed within semi-supervised learning framework, one of the most common is the self-learning algorithm. The self-learning algorithm begins by training the model with labeled samples. Once the training process is completed, the model calculates probability estimates for the unlabeled samples. These estimated probabilities are then compared to a predefined threshold value. Samples with probability values exceeding the threshold are assigned pseudo-labels. The samples, which were labeled, are removed from the set of unlabeled samples and added to the labeled samples. In the next iteration, the model is retrained from scratch using the updated set of labeled samples. This iterative process continues until no unlabeled samples remain. In the proposed method, the self-learning algorithm has been enhanced by incorporating a ensemble learning approach. Instead of relying solely on the current model's predictions for pseudo-label generation, the method also integrates ensemble predictions from models trained in previous iterations. This ensemble approach was inspired by the velocity vector formed by past gradients in the

Stochastic Gradient Descent with Momentum (SGDM) algorithm. At the end of each iteration, the probability predictions of the trained model on the unlabeled data are stored. For a particular iteration moment, the final result is obtained by combining both the probability estimates from the current model and the ensemble result, which were obtained by using stored probability estimates from previous iterations. When computing the ensemble result from stored probabilities, each model's predictions are weighted according to their accuracy value in the validation dataset. Consequently, the influence of more accurate predictions is increased, while that of less accurate predictions is reduced. During the combing phase, the probability estimates of the current model are weighted using values $(1-0.1), (1-0.2), \ldots, (1-0.9)$, whereas the ensemble result from past models are weighted using values $0.1, 0.2, \ldots, 0.9$. Additionally, an adaptive weighting strategy has been applied. The final probabilities obtained after the combining process are then used for pseudo-label generation. As a result, both the current model and past iterations' models contribute to the pseudo-labeling process. This proposed method introduces an innovation to the self-learning algorithm through a two-stage collective learning approach. The method has been tested with the EfficientNetB0, DenseNet121, and ResNet34 architectures on the CIFAR-10, Intel Image Classification, and GTZAN datasets. The best results were obtained using adaptive weighting. According to experimental findings where all unlabeled samples were labeled, the proposed method achieved an average accuracy improvement of 5.35%–14.82% within the same time frame, depending on the architecture and dataset used. Regarding the highest accuracy observed at the best iteration throughout the process, the proposed method demonstrated an average accuracy improvement of 3.58%–8.81%, depending on the architecture and dataset.

Keywords: Self-learning, ensemble learning, pseudo-labeling, semi-supervised learning

1
GİRİŞ

Bu bölümde, öncelikle yapılan çalışmanın daha iyi anlaşılabilmesi için gözetimli öğrenme (supervised learning), gözetimsiz öğrenme (unsupervised learning) ve yarı gözetimli öğrenme (semi-supervised learning) konularında temel bilgiler verilmiştir. Daha sonra çalışmanın amacı ve altyapısı sunulmuş olup, ardından yarı gözetimli öğrenmenin tarihsel gelişimi ile bu alanda gerçekleştirilen çalışmalar kısaca ele alınmıştır. Son olarak, çalışma kapsamında ortaya atılan araştırma sorularının yer aldığı hipotez bölümü incelenmiştir.

1.1 Tezin Amacı

Geleneksel anlamda makine öğrenmesi yöntemleri, kullanılan veride etiket bilgisinin bulunup bulunmamasına göre gözetimli öğrenme ve gözetimsiz öğrenme olmak üzere iki gruba ayrılır [1]. Gözetimli öğrenmede, örneklemlerin sahip oldukları girdi (input) değerlerine karşılık bir çıkış (output) değeri bulunmaktadır. Temel hedef çıkış değeri bulunan örneklemler ile bir makine öğrenme algoritmasının eğitilip, çıkış değeri bulunmayan örneklemlerin çıkış değerlerini tahmin etmektir. Bu çıkış değerleri bir sınıflandırma problemi (classification) veya sürekli değer tahmini (regression) olabilir. Gözetimsiz öğrenmede ise girdi değerlerine karşılık bir çıktı bulunmamaktadır. Temel hedef ise bu örneklemleri eşleştirmektir. Bir başka deyişle kümeleme (clustring) yapmaktır [2, 3].

Gözetimli öğrenme çalışmalarında ilk başlarda her bir veri kümesine karşılık bir makine öğrenme algoritması eğitilir ve bu algoritma yalnızca eğitildiği veri seti üzerinde çalışırdı. Bu yaklaşım zamanla veri miktarındaki artış ve her veri setine özgü ayrı algoritmanın eğitilmesindeki masraftan dolayı yerini çok büyük veri kümesi ile eğitilmiş modellerin ağırlıklarını kullanmaya evrilmiştir. Böylece transfer öğrenme [4, 5] kavramı geliştirilmiştir. Transfer öğrenme, önceden eğitilmiş (pre-trained) ağırlıklar alınarak kullanılacak olan probleme özgü bir

şekilde ince ayarlama (fine-tuning) yapılır.

Zamanla artış gösteren veri miktarı ile de etiketleme ihtiyacı doğmuştur. Etiketleme işlemi ise hem zaman alıcı hem de masraflı bir iştir. Dolayısıyla gözetimli ve gözetimsiz öğrenmeyi birleştiren yarı gözetimli öğrenme [6, 7] makine öğrenmesinin bir dalı olarak ortaya çıkmıştır. Yarı gözetimli öğrenme algoritmaları genel olarak, öğrenme görevini gerçekleştirirken etiketli verinin yanı sıra etiketsiz veriyi de kullanır. Gözetimli ve gözetimsiz öğrenme arasında bulunan bu yöntem, sayıca daha az etiketli veri ile öğrenmeye başlayıp, etiketi olmayan verileri etiketleyerek devam eder. Böylelikle etiketsiz örneklemlerden yararlanılmış olunur [8].

Yarı gözetimli öğrenme kapsamında zamanla farklı algoritmalar geliştirilmiştir. Bu algoritmalar içersinde en eski ve basit yöntemlerden biri olan kendi kendine öğrenme (self-learning / self-training) algoritmasıdır. Bu algoritma, etiket bilgisi bulanan örneklemler ile kullanılan makine öğrenmesi yönteminin (model) eğitilmesiyle başlar. Eğitimi tamamlanan modelin, etiket bilgisi bulunmayan örneklemler üzerindeki her bir sınıfa ait tahmin sonuçları hesaplanır. Elde edilen tahminler içerisinden güvenilir olan örneklemler etiketli örneklemlere eklenirken etiketsiz örneklemler içerisinden de çıkartılır. Bu sürecin etiketsiz örneklem kalmayana kadar, belli bir iterasyon sayısı kadar veya etiketsiz örneklemlerin belli bir oranı etiketlenene kadar iteratif olarak devam etmesiyle algoritma kurulmuş olur [6, 7].

Yapılan çalışmada ise bu algoritmaya kolektiflik (ensembling) kazandırmak amaçlanmıştır. Bu kapsamda yapay sinir ağlarında kullanılan optimize edici algoritmalardan faydalanılmıştır. Yapay sinir ağlarının öğrenmesi sahip oldukları parametre sayısı kadar boyutlu uzayda rastgele bir noktadan başlamaktadır. Eğitildikleri veri üzerindeki başarımı yüksek bir noktaya (lokal minimum) yakınsayınca da yapay sinir ağlarının öğrenmesi sonlanır. Bu yakınsama sürecinde gradyan düşüm temelli algoritmalar kullanılır. Bu algoritmalardan biri olan SGDM [9] yapılan çalışmada ilham kaynağı olmuştur. Momentumlu Stokastik Gradyan İnişi (SGDM) algoritmasında, modelin sahip olduğu parametreler geri yayılım (back propagation) esnasında hem anlık gradyanlar hem de önceki gradyanlar birlikte kullanılarak güncellenir. Peki bu yaklaşım kendi kendine öğrenme algoritmasında sözde etiket üretimine nasıl adapte edilir? Gradyanlar yerine her iterasyonda eğitilen modelin tahmin olasılıkları yerleştirilebilinir. Bir başka deyişle sözde etiketler üretilirken anlık eğitilmiş modelin kararı ile önceki iterasyonlardan eğitilmiş olan modellerin kararları kullanılabilinir. Tıpkı anlık gradyanlar ile eski gradyanların ortaklaşa kullanılması gibi. Peki bunların oranlarını

belirten momentum değeri ne olmalıdır? Bunun için deneysel çalışmalarda $0,1, 0,2, \ldots, 0,9$ değerleri anlık model ile eski modellerin arasındaki oran olarak belirlenmiştir. Ayrıca, bir diğer optimize edici olan Adam [10] algoritmasından esinlenerek sabit bir oran kullanmak yerine adaptif bir formül tasarlanmıştır. Adam algoritmasında öğrenme oranı (learning rate) nasıl ki her iterasyon (epoch) için adaptif bir şekilde ayarlanıyorsa, önerdiğimiz bu yaklaşımda da anlık model ve eski modeller arasındaki oran her iterasyondan sonra adaptif bir şekilde hesaplanıyor olması incelenmiştir.

Hem sabit oran hem de adaptif oran yaklaşımlarında iki farklı kolektiflik yer almaktadır. Birinci kolektiflik, eski modellerin kendi arasında oluşturdukları sonuçtur. İkinci kollektiflik ise anlık eğitilmiş model ile eski modellerin oluşturmuş oldukları sonuç arasındaki kolektifliktir. Oran değeri ise ikinci kolektiflikte kullanılır. Peki kolektiflik nedir? Aynı problemi çözmek için birden fazla algoritmanın (temel öğreniciler) ortaklaşa karar alma mekanizmasıdır [11]. Bu mekanizma kapsamında temel öğrenicilerin başarımları (bias) ile kararları arasındaki farklılıklar (variance) iki temel kavram olup, birbirlerine genellikle ters oratılıdırlar [12].

Önerilen yöntemi test etmek için Evrişimsel Sinir Ağları (Convolutional Neural Networks) kullanılmıştır. CNN yapıları, birden fazla dizi biçiminde gelen verilerden anlamlı bilgi çıkarmak için tasarlanmışlardır [13]. Bu verilere, üç renk derinliğine sahip ve her renk derinliğinde, görüntünün piksel yoğunluklarını içeren görüntü verileri örnek verilebilir. Ayrıca CNN yapıları, trafik işareti tanıma [14], biyolojik görüntüler üzerinde segmentasyon yapılması [15] ve yüz tespiti [16, 17] gibi çalışmalarda da kullanılmışlardır.

Önerilen yönteme ilişkin ön bulgular [18] çalışmamızda sunulmuştur. Söz konusu çalışmada, önerilen yöntemin her iki kolektif aşamasını hesaplamak için ağırlıksız ortalama kullanılmıştır. Elde edilen deneysel sonuçlar doğrultusunda önerilen yöntem ile aynı zaman dilimi içinde var olan yönteme kıyasla %3,42 daha yüksek doğruluk elde edilmiştir.

1.2 Literatür Araştırması

Gerçek dünya problemleri içerisinde tamamiyle etikeli veri bulmak veya veriyi tamamiyle etiketli hale getirmek fazlaca masraflı ve zaman ihtiyacı olan işlemlerdir. Sürekli artış gösteren veri miktarlarıda göz önüne alındığından bu durumun masrafı aynı hızla artış göstermektedir. Etiketi bulunmayan verilerin doğru bir şekilde etiketlenmesi için ilgili uzmanlar tarafından ön inceleme yapılmalıdır. Bu durum

hem zaman hem de maliyet miktarını arttırmaktadır. Bu sebeplerden dolayı zaman ve insan denetimini, modelin performansını düşürmeden en aza indirmeyi amaçlayan yarı gözetimli öğrenme kavramı ortaya atılmıştır [19].

Yarı gözetimli öğrenme yaklaşımında farklı algoritmalar önerilmiştir. İşbirlikçi öğrenme (Co-training) [20] olarak isimlendirilen algoritmada, öncelikle her biri bir modeli iyi bir şekilde eğitmeye yetecek şekilde veri 2 parçaya bölünür ve 2 farklı sınıflandırıcı bu veri kümeleri ile eğitilir. Her 2 sınıflandırıcının da etiketsiz veri üzerindeki tahminleri alınır ve güvenilir olanlar etiketlendirilir ve bir sınıflandırıcının etiketlemiş olduğu örneklemler ile diğer sınıflandırıcı eğitilir. Bu süreç iteratif bir şekilde tekrarlanarak algoritma kurulur.

Üçlü öğrenme (Tri-training)[21] diye isimlendirilen bir diğer algoritma da ise, 3 model aynı anda etiketli veri kümesinin alt parçaları ile eğitilir ve etiketsiz örneklemler üzerinde her 3 modelinde etiket tahminleri hesaplanır. Herhangi bir örneklem üzerinde 2 model aynı sınıf etiketini vermiş ise bu örneklem, etiketli veri içerisine dahil edilir. Bu süreç iteratif olarak tekrarlanır.

Yapılan çalışmada kullanılan kendi kendine öğrenme algoritması ise yarı gözetimli öğrenme algoritmaları içerisinde en eski ve uygulaması kolay olanıdır. Kendi kendine öğrenme ilk olarak metin verilerinde kelimelerin anlamlarına göre ayrıştırılması için önerilmiştir [22]. İlerleyen süreçte bu algoritma, öznel isimler üzerinden öznellik sınıflandırıcı [23], görseller üzerinden nesne tespiti gerçekleştirmek [24] ve uzaktan algılanmış hiper spektral görüntüler üzerinde sınıflandırma yapılması [25] gibi farklı çalışmalarda kullanılmıştır.

Etiketsiz örneklemlerin etiketlenme aşamasında farklı yaklaşımlar kullanılmaktadır. Sözde etiket (pseudo-label) [26, 27] yaklaşımı bu yaklaşımlardan biridir. Temel olarak eğitimi tamamlanmış modelin etiketsiz veri üzerindeki sınıf tahminleri alınıp, bu tahminler içerisinden tanımlanan olasılık şartından daha yüksek olasılığa sahip tahminlerin etiketleri üretilir. Böylelikle modelin tahminleri arasından en güvenilir örneklemler etiketlenmiş olur. Kullanılan bir diğer yaklaşımda ise veri homojenliğini bozmamak için her bir sınıftan en yüksek tahmin olasılığına sahip k adet (algoritma içerisinde tanımlanır) örneklemin etiket bilgileri üretilmiştir [28]. Her iki yaklaşımda kendi kendine öğrenme algoritmasının sözde etiket üretim aşamasında kullanılmaktadır.

Sözde etiket yaklaşımının başarımını arttırmak için gürültü öğrenci (noisy student) [29] yaklaşımı önerilmiştir. Bu kapsamda öğretmen olarak isimlendirilen model saf etiketli veri (ilk anda doğru etiketleri bulunan örneklemler) ile eğitilir. Sonrasında etiketsiz örneklemler üzerindeki tahminleri doğrultusunda sözde etiketler üretilir.

Öğrenci olarak isimlendirilen model, öğretmen modeline göre daha büyük veya eşit olmak şartıyla tanımlanır. Sonrasında bu öğrenci modeline 3 çeşit gürültü eklenir. Bunlar girdi gürültüsü, model gürültüsü ve stokastik derinliktir. Girdi gürültüsü olarak veri çoğaltma için önerilmiş olan Randaugment [30] kullanılmıştır. Model gürültüsü için ise derin öğrenmelerde ezberlemeyi (overfitting) engellemek için önerilen Dropout [31] kullanılmıştır. En son olarak gradyan yok oluşunu (gradyan vanishing) engellemek için önerilmiş olan stokastik derinlik [32] kullanılmıştır.

1.3 Hipotez

Bu çalışmada incelenen önerilen yaklaşım, geçmiş iterasyonlarda eğitilen modellerin tahminlerini de kullanarak kendi kendine öğrenme algoritmasına yenilik katmaktadır. Fazladan herhangi bir modelin eğitilmesine ihtiyaç duymadan geçmiş modellerden oluşturduğu kolektiflik sayesinde önerilen yaklaşımın, daha yüksek başarımlar elde etmesi beklenmektedir. Ancak bu yeni yaklaşım çeşitli soruları beraberinde getirmektedir. Bunlar arasında, geçmiş iterasyonlardaki modellerin tahminlerinin nasıl birleştirilmesi gerektiği ve geçmiş tüm modellerin kullanılıp kullanılmaması yer almaktadır. Ayrıca, anlık eğitimi tamamlanan model ile önceki modellerin oluşturduğu kolektif sonucun nasıl birleştirilmesi gerektiği önemli bir konudur. Son olarak, önerilen yöntemin zaman gereksinimi mevcut yöntemle karşılaştırıldığında nasıl bir fark ortaya koymaktadır?

2
ÖNERİLEN YÖNTEM

Bu bölümde, öncelikle kendi kendine öğrenme algoritması, önerilen yöntemin tanımı ve akış şeması verilmiştir. İlerleyen kısımlarda ise önerilen yöntemin sahip olması gereken bağımsızlar, kullanılan değişkenler, algoritmaların kaba kodu ve kısıtlar verilmiştir.

2.1 Kendi Kendine Öğrenme

Yarı gözetimli kendi kendine öğrenme algoritmasında, her iterasyon sonunda etiketli veri kullanılarak eğitilen modelin, etiketsiz veri üzerindeki sınıf olasılıkları hesaplanır. Hesaplanan sınıf olasılıkları içerisinden önceden tanımlanmış olan olasılık şartını sağlayan örneklemlerin sözde etiketleri üretilir. Sözde etiket bilgisi bulunan örneklemler etiketli veriye dahil edilirken, etiketsiz veriden çıkartılır. Bu sürecin iteratif bir şekilde devam etmesiyle algoritma kurulmuş olur. İteratif süreci sonlandıran durumlar ise: etiketsiz verinin tükenmesi veya belli bir iterasyon sayısının bulunmasıdır. Çalışma boyunca bu algoritmadan var olan yöntem olarak bahsedilmiştir.

2.2 Yöntemin Tanımı

Yapılan çalışmada ise, 2.1'de tanımı yapılan bu algoritmaya kolektiflik kazandırmak hedeflenmiştir. Bu kapsamda: SGDM algoritmasından yararlanılanarak 2 aşamalı kolektiflik kurulmuştur. Birinci aşama, geçmiş iterasyonlarda eğitilmiş olan modellerin etiketsiz veri üzerindeki sınıf olasılıklarının birleştirilmesidir (ϕ_E). Bu aşama SGDM'deki geçmiş gradyanların oluşturduğu velocity (v_t) vektörüne benzemektedir. İkinci aşamda ise anlık eğitimi tamamlanmış modelin etiketsiz veri üzerindeki sınıf olasılıkları (ϕ_M) ile birinci aşamadan elde edilen olasılıkların birleştirilmesidir (ϕ). Bu durum ise SGDM algoritmasında parametre güncelleştirilmesinde kullanılan v_{t+1} vektörünün elde edilmesine benzemektedir. Bir başka deyişle her iterasyon sonunda sözde etiketler üretilirken, hem o andaki

modelin sınıf olasılıkları hem de önceki iterasyonlardan elde edilen sınıf olasılıkları birlikte kullanılmıştır. Yapılan çalışma boyunca her iterasyon sonunda elde edilen sınıf olasılıkları, sonraki iterasyonlar da kullanılmak için kaydedilmiştir. Önerilen kolektif yaklaşımında sözde etiketler aşağıdaki sırayla elde edilmiştir:

- İlgili iterasyonda eğitilen modelin etiketsiz veri üzerindeki sınıf olasıkları hesaplanır (ϕ_M).
- Geçmiş iterasyonlardan kaydedilmiş olan sınıf olasılıklarının kolektif sonucu (ϕ_E) bulunur.
- ϕ_M ve ϕ_E sonuçlarını kullanarak ϕ hesaplanır.
- Elde edilen nihai ϕ olasılıkları ile tanımlanan olasılık şartı karşılaştırılır ve olasılık şartını aşan örneklemlerin sözde etiketleri üretilir.
- Sözde etiketleri üretilen örneklem sayısının, etiketsiz örneklem sayısına oranı istenilen değere ulaşana kadar ilk dört aşamanın tekrar edilmesi.
- İstenilen oran değer aşıldıktan sonra olasılık şartına bakılmaksızın kalan etiketsiz örneklemlerin sözde etiketleri üretililir.

2.2.1 ϕ_E Olasılıklarının Hesaplanması

Geçmiş iterasyonlarındaki modellerin kolektif sonucu olan ϕ_E aşağıdaki gibi hesaplanmıştır:

- Her zaman son yedi modelin olasılıkları kullanılmıştır. Eğer geçmiş olasıkların sayısı yediden az ise bütün geçmiş olasılıklar kullanılmıştır.
- Her bir geçmiş olasılık, doğrulama verisi üzerindeki doğruluk değeri ile çarpılır. Böylece daha başarılı olan olasılıkların etkisi arttırılmışken, düşük performans gösteren olasılıkların etkisi düşürülmüştür.
- Çarpım ile elde edilen yeni olasılıklar kümülatif olarak toplanır. Toplam sonucu, her olasığın doğrulama verisindeki doğruluk değerlerinin toplamına bölünerek 0-1 arasına normalize edilir.

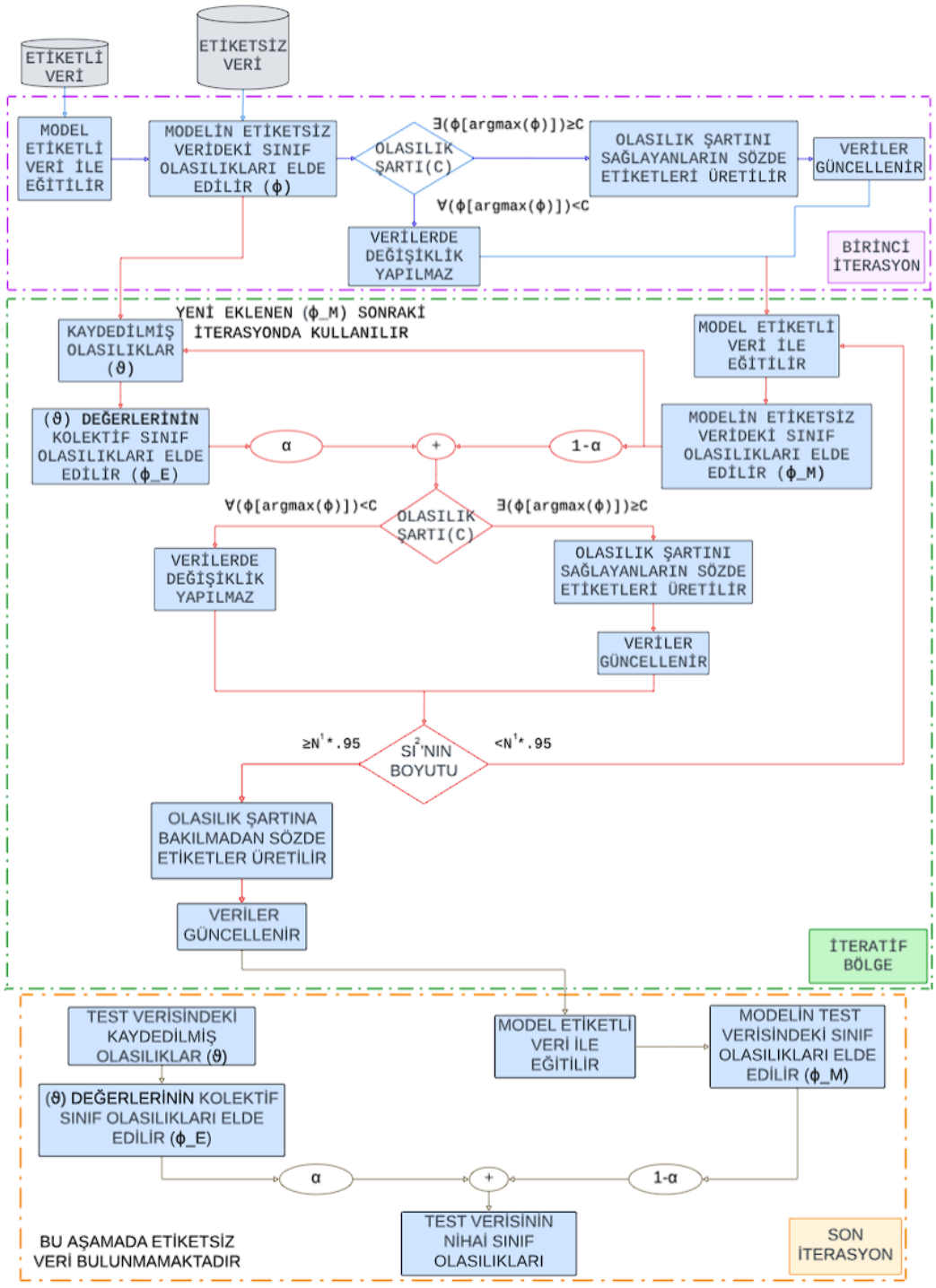
2.2.2 ϕ Olasılıklarının Hesaplanması

Sözde etiket üretiminde kullanılan nihai olasıkları (ϕ) Denklem (2.1) kullanılarak hesaplanmıştır.

$$\phi = (1 - \alpha) * \phi_M + \alpha * \phi_E \tag{2.1}$$

α değeri 0,5 alındığında ϕ_M ve ϕ_E' nin ağırlıksız ortalama hesaplanmış olur. α değeri 0,5'den farklı olduğunda ise ϕ_M ve ϕ_E ağırlıklandırılmış olur.

Önerilen yöntemin akış şeması Şekil 2.1'de verilmiştir.



Şekil 2.1 Önerilen yaklaşımın akış şeması.

[1]N: Başlangıç durumundaki etiketsiz örneklem sayısıdır.

[2]SI: Sözde etiketleri üretilen örneklemelerin konum bilgilerini tutan dizidir.

2.3 Her İterasyondaki Ekstra Çıktılar

Önerilen yöntemin her iterasyondaki davranışını inceleyebilmek için: eğitimi tamamlanmış modelin üç veri kümesi (test, doğrulama ve etiketsiz verileri) için ayrı ayrı ϕ_M, ϕ_E ve ϕ her iterasyon için hesaplanır. Etiketsiz verilerdeki sonuçlar sözde etiket üretiminde, kalanlar ise performans karşılaştırmasında kullanılmışlardır.

2.4 Bağımsızlıklar

Önerilen yöntemin üç farklı durumundan bağımsız olması gereklidir:

- Veri Kümesindeki Örneklemlerden Bağımsızlık: Eğitim verisindeki örneklemlerin test verisindeki örneklemlere benzerlik/farklılıklarına göre modellerin sonuçlarında dalgalanmalar oluşur. Bu durumdan dolayı önerilen yöntemin veri parçalarındaki örneklemlerden bağımsızlığı ispatlanmalıdır.
- Veri Kümesinden Bağımsızlık: Modeller eğitildikleri veri kümesine göre birbirlerine kıyasla farklı sonuçlar verebilir. Yani aynı model bir veri kümesinde çok iyi sonuç verirken farklı bir veri kümesinde kötü sonuç verebilir. Bu durum göz önüne alındığında önerilen yöntemin veri kümesinden de bağımsız olması gerekmektedir.
- Kullanılan Mimariden Bağımsızlık: Derin ağ mimarilerinin sahip oldukları rastgelelikten dolayı aynı veri kümesi ve mimari kullanılmasına (aynı hiper parametreler ile) rağmen sonuçlar arası hafif dalgalanmalar oluşabilir. Buna karşın farklı derin ağ mimarilerinde ise birbirlerine kıyasla sonuçlarda aşırı farklılık görülebilir.

2.5 Neden Olasılıklar Kaydediliyor?

Önerilen yöntemde her iterasyonda sonunda modelin kendisi de kaydedilebilir. Ancak, bu durumda ϕ_E hesaplanırken kaydedilen her bir modelin tekrardan etiketsiz veri üzeriden geçmesi gerekir. Bu durum ilerleyen iterasyonlar ile zamansal yük oluşturur. Bundan dolayı yapılan çalışmada modelin sınıf olasılıkları kaydedilmiştir. Bu durumda ise etiketsiz verinin boyutu değişmemelidir. Sözde etiketleri üretilen örneklemlerim etiketsiz verideki konum/indeks bilgileri kaydedilir. Böylelikle sonraki iterasyonlarda kaydedilmiş olan indekslerin tekrardan sözde etiketleri üretilmemiş olur. Dikkat edilmesi gereken nokta ise: etiketsiz veri sayısı değişmediğinden her iterasyonda tekrardan %70'lik verinin (etiketsiz örneklemlerin oranı) üzerinden geçilmesidir. Bu zamansal olarak yük oluşturmasına rağmen modellerin kaydedilmesi durumuna göre daha hafiftir.

2.6 Değişkenler ve Algoritmalar

Deneysel çalışmalar boyunca kullanılan global değişkenler Tablo 2.1'de verilmiştir. Önerilen yaklaşımda kullanılan fonskiyonların görevleri de Tablo 2.2'de verilmiştir.

Tablo 2.1 Global Değişkenlerin Tanımları

Sembol	Kullanım Amacı	Veri Tipi	Başlangıç Değeri
L_I	Etiketli örneklemlerin resim bilgilerini (piksel değerlerini) tutar	torch.tensor	(%10, 3, Yükseklik, Genişlik)
L_L	Etiketli örneklemlerin etiket bilgilerini tutar	torch.tensor	(%10)
U_I	Etiketsiz örneklemlerin resim bilgilerini tutar	torch.tensor	(%70, 3, Yükseklik, Genişlik)
UD	Etiketsiz veriyi tutar	DataLoader	%70
TD	Test verisini tutar	DataLoader	%10
VD	Doğrulama verisini tutar	DataLoader	%10
N	Etiketsiz verideki örneklem sayısıdır	int	%70
SI	Sözde etiketleri üretilen örneklemlerin konumlarını tutar	np.array	Boş
ϕ_T	Her iterasyonda eğitilen modelin test verisindeki sınıf olasılıklarını tutar	list	Boş
ϕ_V	Her iterasyonda eğitilen modelin doğrulama verisindeki sınıf olasılıklarını tutar	list	Boş
ϕ_U	Her iterasyonda eğitilen modelin etiketsiz verideki sınıf olasılıklarını tutar	list	Boş
C	Olasılık şartı	float	0.995
T_R	Önerilen yöntemin test verisindeki doğruluk sonuçlarını tutar	list	Boş
V_R	Önerilen yöntemin doğrulama verisindeki doğruluk sonuçlarını tutar	list	Boş
V_{MR}	Eğitimi tamamlanmış modelin, doğrulama verisindeki doğruluk değerlerini tutar	list	Boş

Tablo 2.2 Fonksiyonlar ve Görevleri

Fonksiyon İsmi	Görevi
make-torch-dataset	Modelin eğitiminde kullanılacak olan etiketli veriyi uygun hale getirir. Bunun için L_I ve L_L torch.tensor yapılarını birleştirerek, tek bir DataLoader yapısı haline getirir
obtain-current-probabilities	Her iterasyonda eğitimi tamamlanmış modelin; TD, VD ve UD verileri üzerindeki sınıf olasılıklarını hesaplamak için kullanılır. Hesaplanan bu olasılıklar sırasıyla ϕ_T, ϕ_V ve ϕ_U'a eklenir. Sonrasında ϕ_E hesaplanırken ihtiyaç duyulan doğrulama verisindeki doğruluk değerini, ϕ_V[-1]'i kullanılarak hesaplayıp, V_{MR}'a ekler.
val-test-scores-per-iteration	Önerilen yöntemin test ve doğrulama verilerindeki sınıf olasılıklarını kullanarak doğruluk değerlerini hesaplar. Hesaplanan test doğruluk değerini T_R'a, doğrulama doğruluk değerini ise V_R'a ekler.
update-indexes	Önerilen yöntemin etiketsiz veri üzerindeki olasılıklarını, olasılık şartı ile karşılaştırır. Olasılık şartını sağlayan örneklemleri L_I'a, sözde etiketlerini ise L_L'e ekler. Bu örneklemlerin etiketsiz verideki konum bilgilerini ise SI'a ekler.
obtain-probabilities	Bir modelin, bir veri üzerindeki sınıf olasılıklarını hesaplamak için kullanılır.
obtain-ensemble-probabilities	Önerilen yöntemin etiketsiz, test ve doğrulama verilerindeki sınıf olasılıklarını bulmak için kullanılır. İhtiyaç duyulan ϕ_M olasılıkları; ϕ_T, ϕ_V ve ϕ_U değişkenlerinin son değerleridir. ϕ_E ise fonksiyon içerisinde hesaplanır. Bunun için etiketsiz, test ve doğrulama verilerindeki örneklem sayısı kadar satıra, sınıf sayısı kadar kolona sahip sıfır matrisi tanımlanır. Sonrasında hesaplama işlemlerinde bu matrisler kullanılır.

Önerilen yaklaşımın sözde kodu Algoritma 1'de verilmiştir. Tablo 2.2 verilen fonskiyonların sözde kodları ise Algoritma 2 ve Algoritma 7 arası Tablo 2.2'deki sırayla verilmiştir.

Algorithm 1 Önerilen Yöntem

Ara Değişkenler: Test kolektif olasılıklar(tko), Doğrulama kolektif olasılıklar(dko), etiketsiz kolektif olasılıklar(eko), Model(Θ), Etiketli veri(LD)

iterasyon $\leftarrow 1$;
while *len(*SI*) $\leq 0,95 * N$* **do**
  $LD \leftarrow$ make-torch-dataset();
  Θ tanımlanır ve LD ile eğitilir;
  obtain-current-probabilities(Θ,False);
  if *len(*SI*) = 0* **then**
    val-test-scores-per-iteration(ϕ_T[-1],ϕ_V[-1]);
    update-indexes(ϕ_U[-1],True,False);
  else
    $tko, dko, eko \leftarrow$ obtain-ensemble-probabilities(0.5,False);
    val-test-scores-per-iteration(tko, dko);
    update-indexes(eko,False,False);
  end
  if *iterasyon < 3* **then**
    $C \leftarrow C$ - 0,01;
  else if *iterasyon $\geq$ 3 and iterasyon < 5* **then**
    $C \leftarrow C$ - 0,015;
  else
    $C \leftarrow C$ - 0,02;
  end
  iterasyon $\leftarrow$ iterasyon $+ 1$;
end
update-indexes(eko,False,True);
$LD \leftarrow$ make-torch-dataset();
Θ tanımlanır ve LD ile eğitilir;
obtain-current-probabilities(Θ,True);
$tko, dko \leftarrow$ obtain-ensemble-probabilities(0.5,True);
val-test-scores-per-iteration(tko, dko);

Algorithm 2 obtain-current-probabilities

Girdiler: Model(Θ), son $\leftarrow False$

global VD,TD,UD,ϕ_T,ϕ_V,ϕ_U,V_{MR};
$\phi_T \leftarrow \phi_T \oplus^1$ obtain-probabilities(Θ,TD);
$\phi_V \leftarrow \phi_V \oplus$ obtain-probabilities(Θ,VD);
$V_{\text{MR}} \leftarrow V_{\text{MR}} \oplus \phi_V$[-1]'in doğruluk değeri;
if *son = $False$* **then**
  $\phi_U \leftarrow \phi_U \oplus$ obtain-probabilities(Θ,UD);
end
return;

Algorithm 3 make-torch-dataset

Çıktılar: Etiketli veri(EV)

global L_I,L_L;
$EV \leftarrow$ TensorDataset(L_I,L_L);
$EV \leftarrow$ DataLoader(L_I,L_L);
return EV;

Algorithm 4 val-test-scores-per-iteration

Girdiler: Test olasılık(to), Doğrulama olasılık(do)

global T_R,V_R;
$T_R \leftarrow T_R \oplus to$'nun doğruluk değeri;
$V_R \leftarrow V_R \oplus do$'nun doğruluk değeri;
return;

Algorithm 5 update-indexes

Girdiler: Etiketsiz verideki olasılıklar(ϕ), ilk $\leftarrow$ False, son $\leftarrow$ False
Ara Çıktılar: İndexler(γ), Sözde etiketler(β)

global U_I,L_I,L_L,SI,C;
if *son* **then**
    $\gamma \leftarrow$ 0'dan başlayıp N'e kadar birer artışla index dizisi oluşturulur;
    $\gamma \leftarrow \gamma \ominus^2 SI$;
    $\beta \leftarrow argmax(\phi[\gamma])$ ile sözde etiketler üretilir;
    $L_I \leftarrow L_I \oplus U_I[\gamma]$;
    $L_L \leftarrow L_L \oplus \beta$;
    $SI \leftarrow SI \oplus \gamma$;
    return;
end
$\gamma \leftarrow max(\phi) \geq C$ şartını sağlayanların indexleri bulunur;
if *ilk = False* **then**
    $\gamma \leftarrow \gamma \ominus SI$;
end
if *len(γ) = 0* **then**
    return;
end
$\beta \leftarrow argmax(\phi[\gamma])$ ile sözde etiketler üretilir;
$L_I \leftarrow L_I \oplus U_I[\gamma]$;
$L_L \leftarrow L_L \oplus \beta$;
$SI \leftarrow SI \oplus \gamma$;
return;

Algorithm 6 obtain-probabilities

Girdiler: Model(Θ), Veri(I)
Çıktılar: Sınıf olasılıkları(ϕ)

$\phi \leftarrow$ Softmax(Θ'nın I üzerindeki sonuçları);
return ϕ;

Algorithm 7 obtain-ensemble-probabilities

Girdiler: α, son $\leftarrow$ False

Ara Çıktılar: Anlık test olasılıkları(ato), Anlık doğrulama olasılıkları(ado), Anlık etiketsiz olasılıkları(aeo), kolektif test olasılıkları(kto), kolektif doğrulama olasılıkları(kdo), kolektif etiketsiz olasılıkları(keo), Boyut(boyut), Toplam(toplam)

Çıktılar: Nihai test olasılıkları(nto), Nihai doğrulama olasılıkları(ndo), Nihai etiketsiz olasılıkları(neo)

global ϕ_T, ϕ_V, ϕ_U, V_{MR};
ato $\leftarrow$ ϕ_T[-1];
ado $\leftarrow$ ϕ_V[-1];
aeo $\leftarrow$ ϕ_U[-1];
kto $\leftarrow$ (Test örneklem sayısı, sınıf sayısı) boyutlu 0 matrisi tanımlanır;
kdo $\leftarrow$ (Doğrulama örneklem sayısı, sınıf sayısı) boyutlu 0 matrisi tanımlanır;
keo $\leftarrow$ (Etiketsiz örneklem sayısı, sınıf sayısı) boyutlu 0 matrisi tanımlanır;
boyut $\leftarrow$ len(ϕ_T) - 1;
if *boyut* $\geq$ *7* **then**
    toplam $\leftarrow$ V_{MR}[-8:-1] değerleri toplanır;
    for *index* $\leftarrow 0, 1, \ldots, 6$ **do**
        kto $\leftarrow$ kto + V_{MR}[index-8] * ϕ_T[index-8];
        kdo $\leftarrow$ kdo + V_{MR}[index-8] * ϕ_V[index-8];
    end
    kto $\leftarrow$ kto / toplam;
    kdo $\leftarrow$ kdo / toplam;
    nto $\leftarrow$ (1-α) * ato + α * kto;
    ndo $\leftarrow$ (1-α) * ado + α * kdo;
    if *son* **then**
        return nto, ndo
    else
        for *index* $\leftarrow 0, 1, \ldots, 6$ **do**
            keo $\leftarrow$ keo + V_{MR}[index-8] * ϕ_U[index-8];
        end
        keo $\leftarrow$ keo / toplam;
        neo $\leftarrow$ (1-α) * aeo + α * keo;
        return nto, ndo, neo
    end
else
    total $\leftarrow$ V_{MR}[:boyut] değerleri toplanır;
    for *index* $\leftarrow 0, 1, \ldots,$*boyut* **do**
        kto $\leftarrow$ kto + V_{MR}[index] * ϕ_T[index];
        kdo $\leftarrow$ kdo + V_{MR}[index] * ϕ_V[index];
        keo $\leftarrow$ keo + V_{MR}[index] * ϕ_U[index];
    end
    kto $\leftarrow$ kto / toplam;
    kdo $\leftarrow$ kdo / toplam;
    keo $\leftarrow$ keo / toplam;
    nto $\leftarrow$ (1-α) * ato + α * kto;
    ndo $\leftarrow$ (1-α) * ado + α * kdo;
    neo $\leftarrow$ (1-α) * aeo + α * keo;
    return nto, ndo, neo
end

Fonksiyonlarda görülen özel durumlar aşağıdaki gibi ele alınmıştır:

- obtain-current-probabilities: Son iterasyonda bütün sözde etiketler üretildiğinden ϕ_U'nun hesaplanmasına ihtiyaç yoktur. Bu sebepten dolayı son durum fonksiyona girdi olarak verilip, *if* bloğu ile kontrol edilmiştir.

- update-indexes: Birinci iterasyonda SI boş olduğundan $\gamma \leftarrow \gamma \ominus SI$ işlemi hata verecektir. Bu sebeple ikinci *if* bloğu ile birinci iterasyon kontrol edilmiştir. Son iterasyonda ise olasılık şartına bakılmaksızın sözde etiketler üretilmelidir. Bu aşamada 0'dan N'e kadar dizi oluşturulur ve bu dizi ile SI'ın farkı ($\gamma \leftarrow \gamma \ominus SI$) alınır. İşlem sonucunda elde edilen dizi içerisinde sözde etiketleri üretilmemiş olan örneklemlerin konum bilgileri yer alacaktır ve bu konumlardaki örneklemlerin sözde etiketleri üretilir. Son durum birinci *if* bloğu ile kontrol edilmiştir. Ortaya çıkan bu iki özel hususdan dolayı ilk ve son foksiyona girdi olarak verilmiştir. Herhangi bir örneklemin olasılık şartını sağlamaması durumda ise verilerde değişiklik yapılmamalıdır. Bu sebeple üçüncü *if* bloğu kullanılmıştır.

- obtain-ensemble-probabilities: ϕ_E sınıf olasılıkları, kaydedilmiş olasılıklar arasından son yedi olasılık değerleri üzerinden hesaplanır. Bu sebeple kaydedilmiş olasılıkların boyutu *if-else* bloğu ile kontrol edilmiştir. Sekizinci iterasyondan önce *else* kısmı çalışırken, ileriki iterasyonlarda ise *if* kısmı çalışmaktadır. Son iterasyonda ϕ_U hesaplanmayacağından dolayı bu durum *if* kısmı içerisinde ayrıca ele alınmıştır. Bu sebeple fonksiyona son girdi olarak verilmiştir. *Else* kısmında son durumun olmamasındaki sebep: toplam iterasyon sayısının, sekizden fazla olacağının garantisidir. Ayrıca *obtain-ensemble-probabilities* fonksiyonu çağrılmadan önce *obtain-current-probabilities* fonksiyonu çağrıldığından: ϕ_T, ϕ_V ve ϕ_U değerlerinin sondaki olasılıkları anlık iterasyonda eğitilmiş olan modele aittir. Bu sebeple ϕ_E hesaplanırken asla ϕ_T, ϕ_V ve ϕ_U'nun sondaki değerlerine gidilmemelidir. Bu durumu *if* içerisinde sağlarken *for* döngüsü $index \leftarrow 0, 1, \ldots, 6$ şeklinde tanımlanmış olup, [index-8] konum bilgisiylede sondan ikinci değere kadar erişilmiştir. *Else* kısmında ise for döngüsü $index \leftarrow 0, 1, \ldots,$boyut şeklinde tanımlanmıştır ve boyut değeri len(ϕ_T) - 1 olarak alındığından [index] konum bilgisi asla son elemana erişemeyecektir. Böylece ϕ_E değeri doğru bir şekilde hesaplanacaktır.

[1] $\oplus$:Sol taraftaki değişken içerisine sağ tarafta verilen değişkeni ekler.
[2] $\ominus$:Sol taraftaki değişken içerisine sağ tarafta verilen değişkeni çıkartır.

2.7 Kısıtların Tanımlanması

Yapılan çalışmada olasılık şartında azalış yapılması, akışın yüzdelik (SI boyutunun N' oranı) şekilde sonlandırılması ve son yedi modelin kullanılması kısıtlar olarak tanımlanmıştır. Tanımlanmış bu kısıtların detaylı incelemesi dördüncü bölümde yer almaktadır. Tablo 2.3'de veri kümelerine ve yönteme göre alınan kısıtlar verilmiştir.

Tablo 2.3 Kısıtlar

Veri	Yöntem	Azalış Miktarı	SI/N Oranı
CIFAR10	Var Olan	Sabit, 0.005, 0.01	0.98
	Önerilen	0.01, 0.015, 0.02	0.95
		0.01, 0.015, 0.02	0.925
Intel	Var Olan	0.005, 0.01	0.98
	Önerilen	0.0125, 0.025	0.95
Gtzan	Var Olan	0.0025, 0.0075	0.98
	Önerilen	0.0075, 0.02	0.95

Tablo 2.3'de bulunan Azalış Miktarı CIFAR10 veri kümesindeki sırasıyla birinci ve ikinci iterasyonlar, üçüncü ve dördüncü iterasyonlar ve beşinci ve sonraki iterasyonlar içindir. İlk iterasyonlarda daha güvenli ilerlemek için bu tercih yapılmıştır. Intel ve Gtzan veri kümeleri için ise sırasıyla birinci ve ikinci iterasyonlar ile geriye kalan iterasyonlar içindir.

CIFAR10 veri kümesindeki önerilen yöntemin üst satırı α'nın $0,1,0,2,0,3,0,4$ ve $0,5$ değerlerinde, alt satırı ise α'nın geriye kalan değerlerinde kullanılan kısıtlardır. α değerinin artmasına paralel olarak nihai sonucun elde edilme süresinde de artış gözlemlendiğinden, var olan yöntemdeki süreyi aşmamak için bu yol izlenmiştir. Deneysel sonuçlar bölümünde sonuçların elde edilmesi için geçen süre verilmiştir.

Yapılan çalışmanın temel amacı var olan yönteme göre daha başarılı sonucu yakın veya daha az sürede elde etmek olduğundan önerilen yöntemde daha fazla risk alınmıştır. Çünkü kısıtların incelenmesi bölümünde açıklanacağı üzere önerilen yöntem aynı şartlar altında daha fazla iterasyona ihtiyaç duymaktadır. Ayrıca önerilen yöntemde her iterasyon sonrasında elde edilen çıktılar daha fazla olduğundan her bir iterasyonu var olan yönteme kıyasla biraz daha fazla sürede sonuç üretebilmiştir.

3
DENEYSEL SONUÇLAR

Bu bölümde öncelikli olarak önerilen yöntemin bağımsızlıkları incelenmiştir. Sonrasında sonuçların anlaşılabilmesi için gerekli açıklamalar yapılmıştır. İleriki kısımlarda ise kullanılan veri setleri tanıtılıp, deneysel sonuçlar verilmiştir.

Bölüm 2'de bahsedilen önerilen yöntemin bağımsızlıkları aşağıdaki gibi sağlanmıştır:

- Veri kümesindeki örneklemlerden bağımsızlık şartını sağlamak için bütün süreçler üç kez çalıştırılmıştır. Her süreçte veri kümesi parçalanmadan önce farklı tohum (seed) değeri ile karıştırılmıştır. Böylece bir süreçteki veri parçalarında bulunan örneklemlerin diğer süreçlerdekinden farklılık göstermesi sağlanmıştır. Sırasıyla 23, 27 ve 42 değerleri tohum olarak kullanılmıştır.

- Veri kümesinden bağımsızlığı ispatlamak için CIFAR10 [33], Intel [34] ve Gtzan [35] veri kümeleri deneysel çalışmalarda kullanılmıştır. Böylelikle önerilen yöntemin farklı veri kümeleri üzerindeki başarımları hesaplanabilinmiştir.

- Modellerin sahip oldukları mimari yapılarından bağımsızlık şartını kanıtlamak için CIFAR10 ve Intel veri setlerinde EfficientNetB0 [36] ve DenseNet121 [37] olmak üzere iki farklı mimari kullanılmıştır. Her iki mimaride de eğitimler yapılırken, önceden eğitilmiş (pre-trained) ağırlıklar kullanılmamıştır. Yani her iterasyonda başlangıç parametreleri rastgele atanmıştır. Gtzan veri seti için ResNet34 [38] kullanılmış olup, bu mimarinin de önceden eğitilmiş ağırlıkları alınmamıştır.

Sonuçların elde edilmesinde kullanılan yöntemler ve anlaşılabilirliği için önemli notlar:

- Tablolarda verilen süre değeri: birinci iterasyonun başlamasından itibaren nihai sonucun elde edilmesine kadar geçen toplam zamanı saniye cinsinden ifade etmektedir.

- Deneysel çalışmalar boyunca modellerin eğitim aşamalarını sonlandıran iki durum vardır. Bunlar: maksimum devir sayısı ve eğitildikleri veri üzerindeki doğruluk değeridir. Maksimum devir sayısı, Intel ve CIFAR10 için 20 olarak ayarlanmış olup, Gtzan için 35 olarak belirlenmiştir. Eğitildikleri veri üzerindeki doğruluk değeri Intel ve CIFAR10 için %96 olup, Gtzan için %99'dur. Her devir için ayrıca doğrulama verisi üzerindeki doğruluk ve hata değeri izlenip, en uygun sonuca ulaşılan devirin parametreleri tekrardan modele aktarılmıştır. Örneğin: eğitim aşamasında doğrulama verisi üzerindeki doğruluk ve hata değerlerine göre en başarılı sonuç yedinci devirde elde edilmiş olsun. Modelin eğitimi tamamlandıktan sonra eğitimi durduran şarttan (ister maksimum devir sayısına ulaşmış olsun ister eğitim verisindeki en yüksek doğruluğuna ulaşmış olsun) bağımsız olarak yedinci devir sonundaki parametreler modele yüklenmiştir. Böylelikle modelin en optimum sonucu vermesi sağlanmıştır. CIFAR10 ve Intel veri kümelerindeki eğitim verisi üzerindeki doğruluk değeri %92'den sonra model ezberlemeye doğru gitmiştir. Bu sebeple eğitimler %96'da sonlandırılmıştır. Gtzan veri setinde ise ezberleme %95'lerden sonra başlamıştır.

- En iyi iterasyon anındaki tablolar oluşturulurken, doğrulama ve test verisi için ayrı ayrı en iyi iterasyon sonuçları bulunmuştur. Doğrulama verisindeki en yüksek başarım, her zaman test verisinde de en yüksek başarıma sahip olmayacağından bu yol izlenmiştir.

- Her iterasyondaki sonuçların ortalama grafikleri elde edilirken, üç tohumun da her bir iterasyondaki sonuçlarının ortalaması alınmıştır. Ancak tohumlar arası toplam iterasyon sayısı farklılık gösterebilmiştir. Bu durumdan dolayı iterasyon sayısı az olanların en son değerleri iterasyon sayısı fazla olanla eşitlenene kadar tekrar edilmiştir.

- Modeller eğitilirken AdamW [39] optimize edici olarak kullanılmıştır. Yapılan bütün deneysel çalışmalarda öğrenme oranı (learning rate) 1e-3 olarak ayarlanmıştır. Geriye kalan parametreler varsayılan olarak kullanılmıştır.

- Sözde etiketleri üretilen örneklemler, etiketli veriye dahil edildikten sonra etiketli veri kendi içerisinde rastgele bir biçimde (sabit bir tohum verilmeden) karıştırılmıştır. Yani her iterasyon sonunda etiketli veriye eklenen

örneklemler sürekli olarak en sonda yer almayıp, karıştırma işlemi ile dağıtılmışlardır.

- Modeller eğitilirken başlangıç parametreleri rastgele atanmıştır. Yani her iterasyonun başlangıç anında modeller diğer iterasyonlardaki parametrelerine göre farklı değerlere sahiptirler.
- Bu bölümde verilen tablolardaki Doğruluk, Kesinlik, Duyarlılık ve F1 sütunlarındaki değerler yüzdelik (%) olarak hesaplanmıştır. Süre sütunundaki değerler ise birinci iterasyonun başlamasından nihai sonuçların elde edilmesine kadar geçen süreyi saniye cinsinden vermektedir.

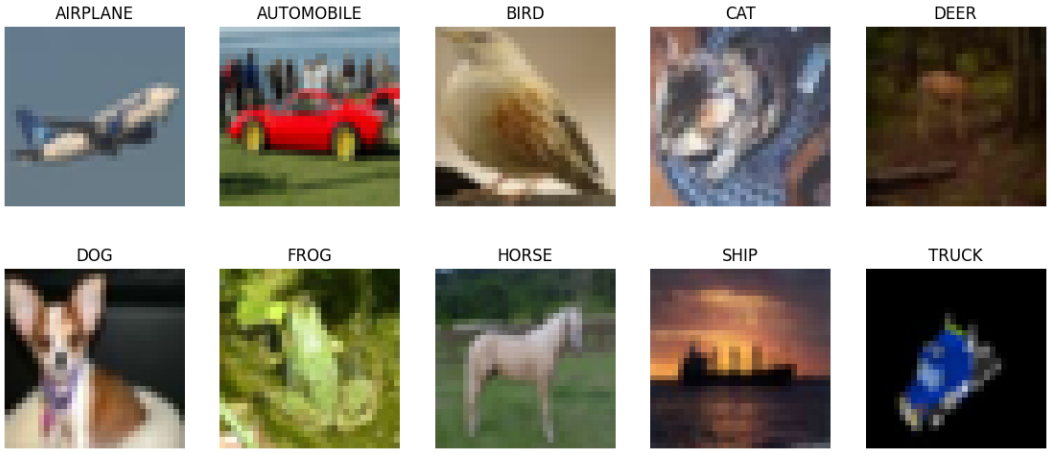
3.1 Veri Setlerinin Tanıtımı

Deneysel çalışmalar boyunca CIFAR10, Intel ve Gtzan veri setleri kullanılmıştır. Veri setlerine ait bilgilere bu kısımda yer verilmiştir.

3.1.1 CIFAR10

CIFAR10 veri seti 10 farklı nesneye ait görüntüler içermektedir. Toplam 50.000 örneklemin bulunduğu veri setinde her sınıf için eşit sayıda (5000 adet) örneklem yer almıştır. Görüntüler 3 boyutlu (RGB) olup, yükseklik ve genişliği sırasıyla 32x32 pikseldir. Her bir nesneye ait görüntü ve nesne ismi Şekil 3.1'de verilmiştir.

Veri ön işleme aşamasında görüntüler, bikübik interpolasyon kullanılarak 84x84 boyutuna getirilip, derinliği korunmuştur. Görüntülerin RGB kanalları normalize edilirken, ortalamaları (0,485, 0,456, 0,406) ve standart sapmaları (0,229, 0,224, 0,225) olacak şekilde ayarlanmıştır. Toplu boyut (batch size) değeri 64 alınmıştır. Son kısımda kalan artık resimler ise kullanılmamıştır.

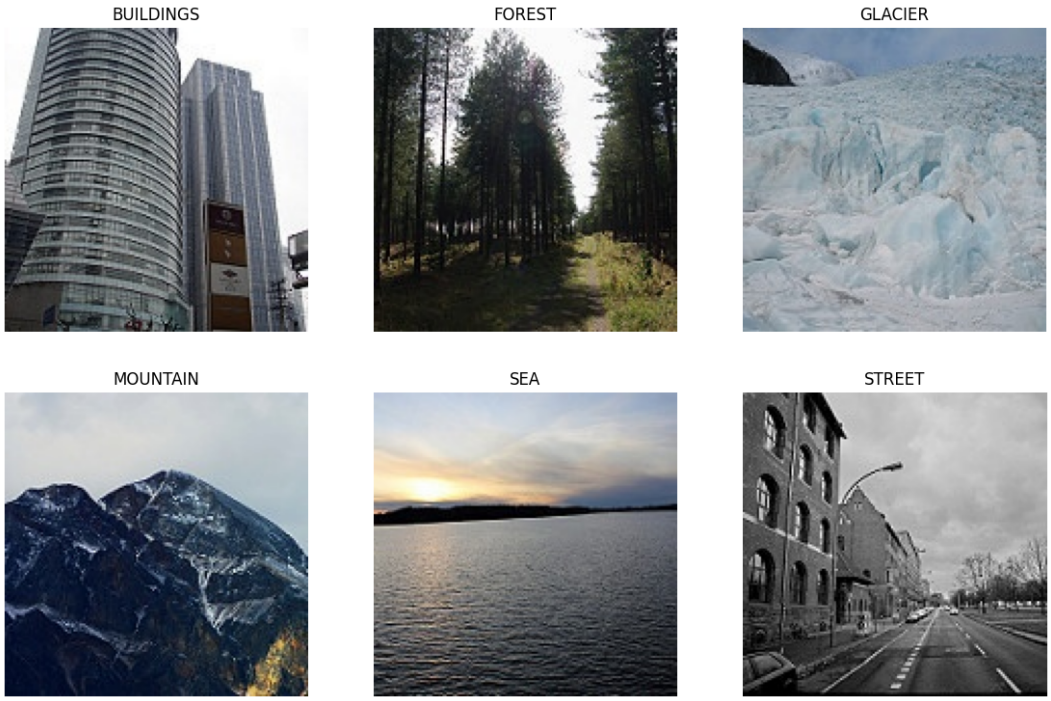


Şekil 3.1 CIFAR10 Veri Seti

3.1.2 Intel

Intel veri seti 6 farklı nesneye ait görüntüler içermektedir. Toplam 17.034 örneklemin bulunduğu veri setinde sınıfların sahip olduğu örneklem sayıları mükemmel homojenlik durumunda bulunmamalarına rağmen birbirlerine yakın sayıda örnekleme sahiptirler. Görüntüler 3 boyutlu (RGB) olup, yükseklik ve genişliği sırasıyla 150x150 pikseldir. Her bir nesneye ait görüntü ve ismi Şekil 3.2'de verilmiştir.

Veri ön işleme aşamasında görüntüler, bikübik interpolasyon kullanılarak 128x128 boyutuna getirilip, derinliği korunmuştur. Görüntülerin RGB kanalları normalize edilirken, ortalamaları (0,485, 0,456, 0,406) ve standart sapmaları (0,229, 0,224, 0,225) yapılmıştır. Etiketli verileri eğitim için uygun hale getirirken toplu boyut değeri 32 alınmıştır. Son kısımda kalan artık resimler ise kullanılmamıştır.



Şekil 3.2 Intel Veri Seti

3.1.3 Gtzan

Gtzan veri seti 10 farklı müzik türüne (blues, classical, country, disco, hiphop, jazz, metal, pop, raggae, rock) ait spectrogram görüntülerini içermektedir. Toplam 1000 örneklemin bulunduğu veri setinde her bir sınıf için 100 örneklem vardır. Görüntüler 3 boyutlu (RGB) olup, yükseklik ve genişliği sırasıyla 288x432 pikseldir.

Gtzan veri seti ayarlanırken aşağıdaki yol izlenmiştir:

- Her bir örneklemde beyaz çerçeve bulunduğundan ve bu çerçeve herhangi bir anlam içermediğinden öncelikle bu çerçeve kaldırılmıştır. Böylelikle örneklemlerin yükseklik ve genişliği sırasıyla 218x336 getirilmiştir.

- Örneklem sayısını çoğaltmak için görüntüler 2 eşit parçaya bölünmüştür. Yani 30 saniyelik olan her bir spectrogram görüntü, 15'er saniye olmak üzere 2 örneklem haline getirilmiştir. Böylelikle toplam 2000 örneklem elde edilmiştir. Her bir görüntünün yükseklik ve genişliği işlem sonrasında sırasıyla 218x168 olmuştur.

- Veri kümesi her bir tohum için ayrı bir şekilde karıştırılıp %80 eğitim, %10 doğrulama ve %10 test için ayrılmıştır. Etiketsiz örneklemler, eğitim kümesi içerisinden seçilmiştir. En son 3 adet veri kümesi elde edilmiştir.

- Her bir veri kümesindeki örneklemler, ayrı ayrı veri çoğaltma katmanından geçirilmiştir. Elde edilen yeni görüntüler de çoğaltıldıkları veri kümesine eklenmiştir. Böylelikle her bir veri kümesi 4,000 örneklem içermiş olur.

Dikkat edilmesi gereken bir diğer nokta ise eğitim kümesi içerisindeki örneklemlerin, doğrulama ve test kümelerinde bulunmaması gereklidir. Örneklemler veri çoğaltma katmanından önce bölündüğü için bu durum sağlanmış olur. Eğer örneklemler en son aşamada bölünmüş olsaydı, eğitim kümesi içerisindeki herhangi bir örneklemin veri çoğaltma katmanından sonraki hali test veya doğrulama kümesi içerisinde bulunabilirdi.

Veri çoğaltma katmanındaki her bir katman *torchvision* kütüphanesi içerisinde bulunan *transforms* modülünden alınmıştır. Veri çoğaltma katmanı aşağıdaki sırayla inşa edilmiştir:

- *RandomRotation* fonksiyonu kullanılarak örneklemler -4 ile +4 arası rastgele döndürülmüştür. *Expand* argümanına *False* verilerek örneklemlerin yükseklik ve genişliğinin koruması sağlanmıştır.

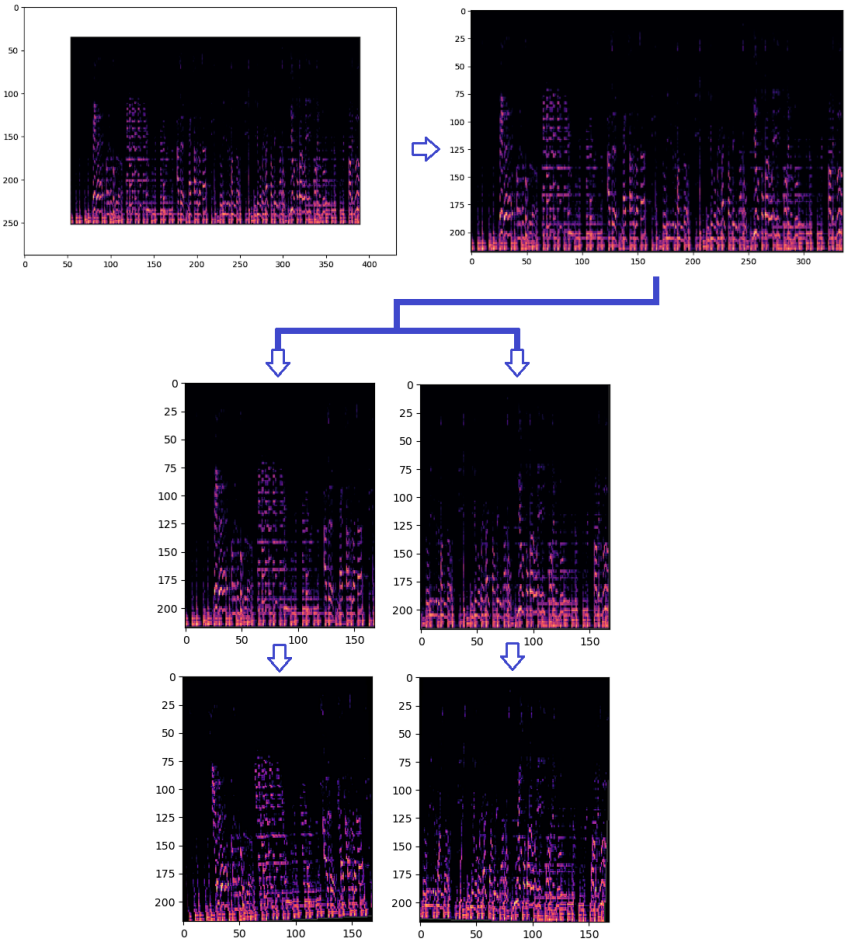
- *RandomHorizontalFlip* fonksiyonu ile yatay eksende simetrisi alınmıştır. Fonksiyonun işlem yapma olasılığına 0,4 girilmiştir.

- *RandomAdjustSharpness* fonksiyonu ile keskinlerştirme yapılmıştır. Keskinleştirme faktörüne 2, işlem yapma olasığına ise 0,4 girilmiştir.

- *ColorJitter* fonksiyonu kullanılarak görüntülerin parlaklık, yoğunluk ve doygunluk değerlerinde rastgele değişimler yapılmıştır. Her bir parametrenin değiştirme etkisi 0,9 ile 1,1 arasındadır.

Veri ön işleme aşamasında görüntü değerleri, 0-1 arasına getirilerek normalizasyon yapılmıştır.

Örneklemlere yapılan değişimler, bir görüntü üzerinden Şekil 3.3'te verilmiştir.



Şekil 3.3 Gtzan Veri Setinin Ayarlanması

3.2 Sonuçlar

Deneysel çalışmalardaki sonuçlar aşağıdaki sırayla elde edilmiştir:

- Veri kümelerinin kullanılan mimarilerdeki genel sonuçlarının elde edilmesi.
- Her bir veri kümesi ve kullanılan mimarilerin var olan kendi kendine öğrenme algoritmasındaki sonuçlarının bulunması.
- Önerilen yöntemdeki α değerinin etkisini incelemek için farklı α değerleri ile önerilen yöntemin sonuçlarının elde edilmesi.
- Sabit α yaklaşımı yerine adaptif α yaklaşımının önerilmesi ve sonuçlarının bulunması.

- Farklı mimari bağımsızlığının incelenmesi.
- Farklı veri seti bağımsızlığının incelenmesi.

3.2.1 Genel Sonuçlar

Genel sonuçlar elde edilirken veri kümeleri: %80 eğitim, %10 doğrulama ve %10 test olmak üzere üç parçaya bölünmüştür. Mimarilerin ulaşılabilecekleri en yüksek doğruluk değerini tespit etmek için bu sonuçlar elde edilmiştir. CIFAR10, Gtzan ve Intel veri kümelerindeki genel sonuçlar sırasıyla Tablo 3.1, Tablo 3.2 ve Tablo 3.3'de verilmiştir.

Tablo 3.1 CIFAR10 Genel Sonuçları

Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1
EfficientNetB0	Test	78,68	78,95	78,89	78,77
		78,78	79,83	78,99	79,27
		79,94	80,03	79,96	79,82
	Doğrulama	80,78	80,96	80,84	80,74
		79,74	80,18	79,66	79,83
		80,62	80,73	80,53	80,42
DenseNet121	Test	81,81	82,65	81,99	82,04
		83,11	83,83	83,35	83,29
		82,03	82,11	82,00	81,90
	Doğrulama	82,15	82,79	82,10	82,18
		83,55	84,01	83,56	83,54
		82,21	82,26	82,11	82,03

Tablo 3.2 Gtzan Genel Sonuçları

Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1
ResNet34	Test	75,50	75,24	74,87	74,52
		77,25	77,36	76,73	76,34
		77,25	77,68	76,39	76,41
	Doğrulama	79,25	78,80	78,24	78,03
		75,75	75,49	76,15	75,40
		76,00	76,26	76,03	75,39

Tablo 3.3 Intel Genel Sonuçları

Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1
EfficientNetB0	Test	85,69	85,80	85,93	85,85
		85,81	86,07	85,96	85,90
		86,53	86,84	86,80	86,72
	Doğrulama	87,50	87,45	87,50	87,47
		87,19	87,38	87,32	87,28
		85,21	85,43	85,44	85,32
DenseNet121	Test	88,64	88,94	88,69	88,79
		89,06	89,40	89,13	89,13
		87,92	88,07	88.20	88,07
	Doğrulama	89,96	90,01	90,00	89,98
		89,18	89,36	89,26	89,31
		87,37	87,55	87,69	87,57

3.2.2 Var Olan Yöntemin Sonuçları

Bu bölümde var olan yöntemin sonuçları verilmiştir. Doğru bir kıyaslama yapmak için sonuca etki edecek faktörlerin: kullanılan mimari yapılar, önceden eğitilmiş ağırlıkların kullanılmaması, veri kümelerinin aynı tohumlar ile karıştırılması, olasılık şartındaki düşüşler, algoritma akışını sonlandıran durma şartı ve kullanılan donanım yapısının sabit kalması gereklidir. Bu etkenler deneysel sonuçları doğrudan etkileyebileceğinden, var olan yöntemin sonuçları ayrıca elde edilmiştir.

Var olan yöntemin CIFAR10, Intel ve Gtzan veri kümelerindeki nihai sonuçları sırasıyla Tablo 3.4, Tablo 3.5 ve Tablo 3.6'da verilmiştir. Her iterasyon sonunda hesaplanmış olan test ve doğrulama verileri üzerindeki doğruluk değerleri ve etiketli veri miktarındaki değişim grafiksel olarak sırasıyla Şekil 3.5, Şekil 3.6 ve Şekil 3.7'de verilmiştir.

Tablo 3.4 CIFAR10 Var Olan Yöntem Sonuçları

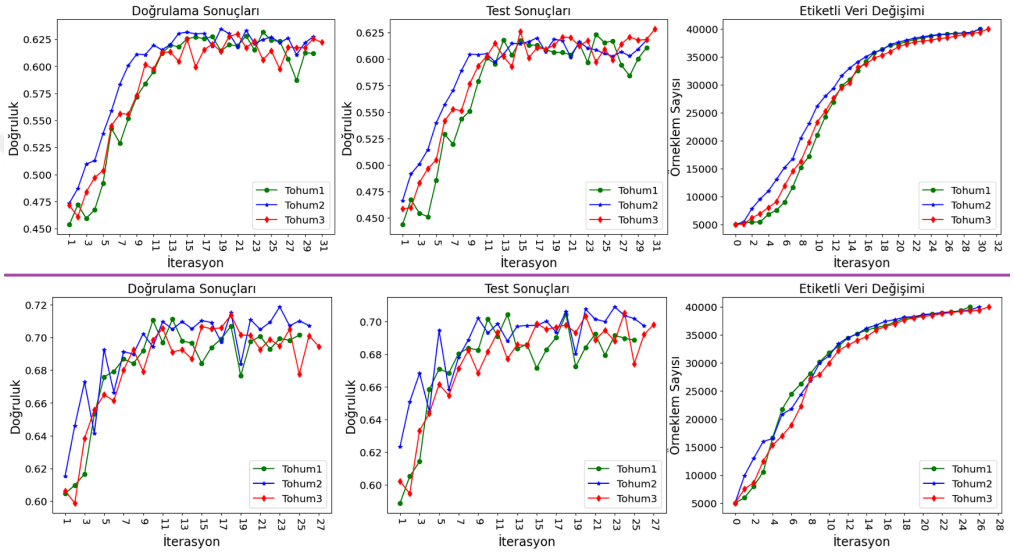
Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
EfficientNetB0	Test	61,07	61,02	61,47	59,99	9860,83
		61,77	62,90	62,33	60,29	10400,62
		62,84	62,17	62,76	61,91	10000,77
	Doğrulama	61,19	61,73	61,30	60,05	9860,83
		62,72	62,96	62,66	60,50	10400,62
		62,19	61,20	62,00	61,04	10000,77
DenseNet121	Test	68,85	68,95	69,16	68,85	9097,54
		69,73	70,05	70,18	69,47	10129,27
		69,81	70,19	69,83	69,77	10453,05
	Doğrulama	70,15	70,36	70,24	70,11	9097,54
		70,73	70,62	70,71	70,17	10129,27
		69,45	69,68	69,29	69,25	10453,05

Tablo 3.5 Intel Var Olan Yöntem Sonuçları

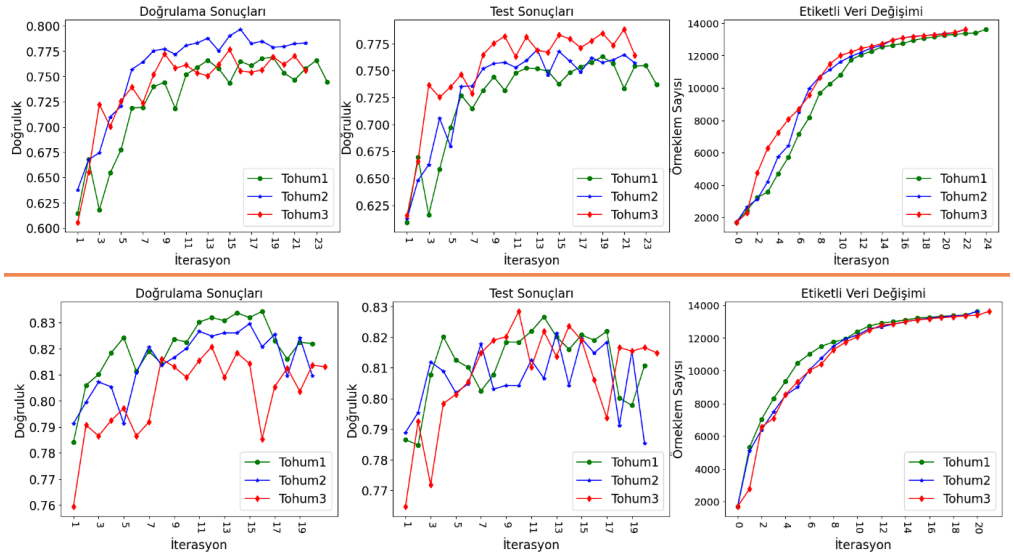
Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
EfficientNetB0	Test	73,70	73,75	73,78	73,69	7878,21
		75,71	75,82	75,90	75,78	7346,41
		76,41	76,32	76,71	76,37	7158,52
	Doğrulama	74,47	74,48	74,40	74,32	7878,21
		78,30	78,31	78,55	78,34	7346,41
		75,59	75,52	75,95	75,62	7158,52
DenseNet121	Test	81,07	81,04	81,34	81,15	7644,30
		78,54	78,69	78,85	78,56	8623,79
		81,49	81,85	81,68	81,71	7694,21
	Doğrulama	82,19	82,15	82,24	82,14	7644,30
		80,95	81,08	81,14	80,99	8623,79
		81,31	81,57	81,51	81,44	7694,21

Tablo 3.6 Gtzan Var Olan Yöntem Sonuçları

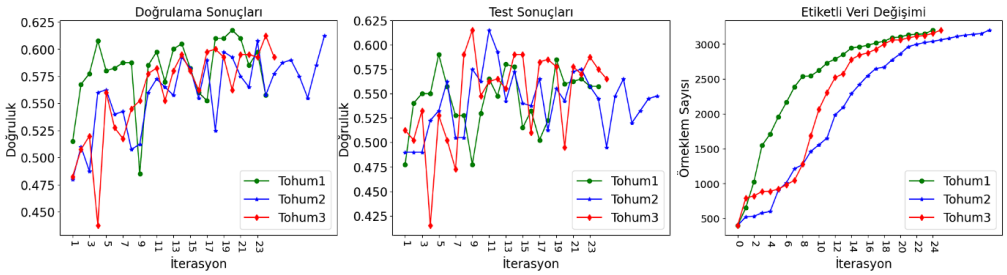
Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
ResNet34	Test	55,75	59,59	55,08	54,92	5083,45
		54,75	52,21	53,77	50,37	5962,89
		56,50	58,38	55,86	54,15	4820,79
	Doğrulama	55,75	59,08	55,89	55,22	5083,45
		61,25	57,05	58,49	56,32	5962,89
		59,25	57,85	58,89	56,95	4820,79



Şekil 3.4 CİFAR10 sonuçları. Üst satır EfficientNetBO, alt satır DenseNet121 mimarisine aittir.



Şekil 3.5 Intel sonuçları. Üst satır EfficientNetBO, alt satır DenseNet121 mimarilerine aittir.



Şekil 3.6 Gtzan sonuçları.

Şekil 3.4, Şekil 3.5 ve Şekil 3.6'daki sonuçlar incelendiğinde, son iterasyonda düşüşler görülebilmiştir. Son iterasyonda olasılık şartına bakılmaksızın sözde etiketler üretildiğinden bu durumun oluşması normaldir. Ayrıca, belli bir iterasyondan sonra doygunluk olmuştur ve sonraki iterasyonlarda kırılmalar yer alabilmiştir.

Yapılan iterasyonlarda doğrulama ve test verilerindeki en yüksek doğruluk değerleri ve iterasyon bilgisi CIFAR10, Intel ve Gtzan veri kümeleri için sırasıyla Tablo 3.7, Tablo 3.8 ve Tablo 3.9'da verilmiştir.

Tablo 3.7 CIFAR10 Var Olan Yöntemin En İyi İterasyon Sonuçları

Mimari	Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
EfficientNetB0	Test	62,32 / 24	61,999 / 17	62,841 / 31
	Doğrulama	63,141 / 24	63,442 / 19	62,981 / 21
DenseNet121	Test	70,433 / 18	70,893 / 23	70,533 / 24
	Doğrulama	71,114 / 12	71,875 / 23	71,374 / 18

Tablo 3.8 Intel Var Olan Yöntemin En İyi İterasyon Sonuçları

Mimari	Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
EfficientNetB0	Test	76,297 / 19	76,946 / 13	78,833 / 21
	Doğrulama	76,887 / 19	79,658 / 16	77,653 / 15
DenseNet121	Test	82,960 / 12	84,316 / 16	82,842 / 10
	Doğrulama	84,434 / 12	85,259 / 21	82,075 / 12

Tablo 3.9 Gtzan Var Olan Yöntemin En İyi İterasyon Sonuçları

Mimari	Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
ResNet34	Test	59,00 / 5	61,50 / 11	61,50 / 9
	Doğrulama	61,75 / 20	61,25 / 31	61,25 / 24

3.2.3 α Değerinin Etkisi

α'nın etkisini gözlemleyebilmek için sırasıyla $0,1,0,2,\ldots,0,9$ değerleri alınmıştır. Bu kısımdaki sonuçlar CIFAR10 veri kümesi ile DenseNet121 mimarisi kullanılarak elde edilmiştir.

3.2.3.1 Önerilen Yöntemin $\alpha = 0,1$ Durumundaki Sonuçları

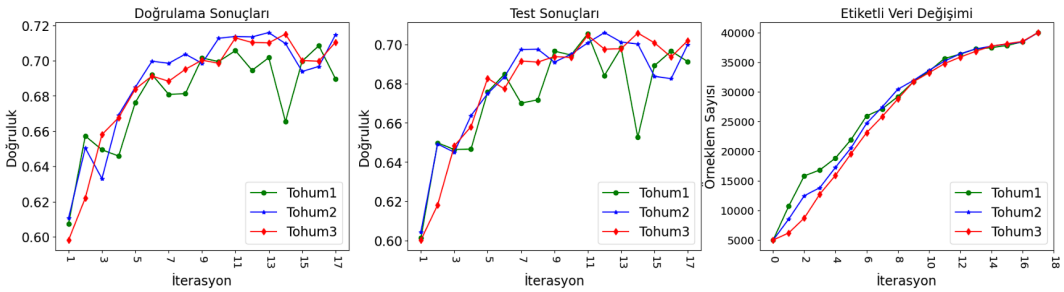
α'nın 0,1 alındığında elde edilen nihai sonuçlar Tablo 3.10'da, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.11'de verilmiştir. Şekil 3.7'de ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.10 α=0,1 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	69,11	69,12	69,44	68,83	6083,48
	69,99	70,21	70,47	69,83	6217,04
	70,19	70,32	70,23	69,95	6064,37
Doğrulama	68,97	68,87	69,03	68,61	6083,48
	71,47	71,35	71,44	70,96	6217,04
	71,05	71,12	70,91	70,72	6064,37

Tablo 3.11 α=0,1 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	70,533 / 11	70,593 / 12	70,573 / 14
Doğrulama	70,853 / 16	71,595 / 13	71,514 / 14



Şekil 3.7 α=0,1 İterasyonların Sonuçları

3.2.3.2 Önerilen Yöntemin $\alpha = 0,2$ Durumundaki Sonuçları

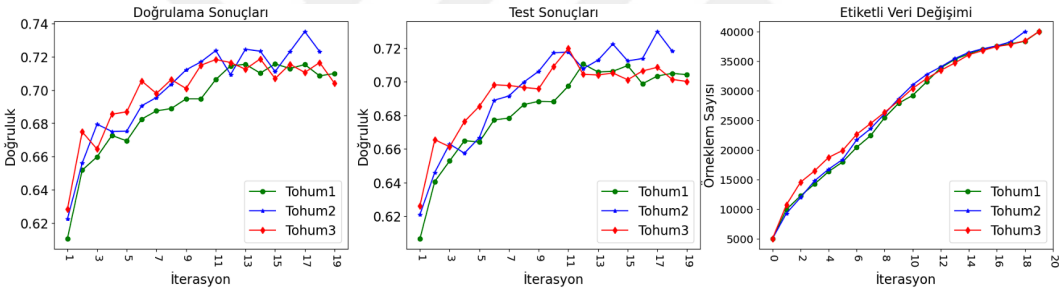
α'nın 0,2 alındığında elde edilen nihai sonuçlar Tablo 3.12'da, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.13'da verilmiştir. Şekil 3.8'de ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.12 α=0,2 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	70,41	70,29	70,71	70,13	6528,50
	71,81	72,01	72,26	71,73	5766,05
	70,01	70,04	70,02	69,90	6567,79
Doğrulama	70,99	71,07	71,09	70,65	6528,50
	72,31	72,36	72,32	72,10	5766,05
	70,41	70,50	70,26	70,25	6567,79

Tablo 3.13 α=0,2 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	71,054 / 12	72,977 / 17	71,975 / 11
Doğrulama	71,575 / 15	73,518 / 17	71,875 / 14



Şekil 3.8 α=0,2 İterasyonların Sonuçları

3.2.3.3 Önerilen Yöntemin $\alpha = 0,3$ Durumundaki Sonuçları

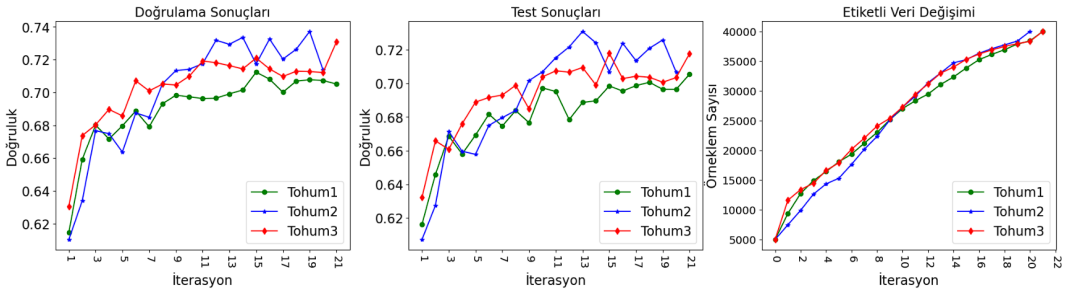
α'nın 0,3 alındığında elde edilen nihai sonuçlar Tablo 3.14'de, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.15'de verilmiştir. Şekil 3.9'da ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.14 α=0,3 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süren(sn)
Test	70,55	70,44	70,88	70,23	7139,59
	70,67	71,23	71,08	70,82	6212,92
	71,77	71,89	71,87	71,61	7134,55
Doğrulama	70,51	70,22	70,54	70,03	7139,59
	71,37	71,54	71,34	71,22	6212,92
	73,07	73,04	72,91	72,67	7134,55

Tablo 3.15 α=0,3 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	70,553/ 21	73,097 / 13	71,795 / 15
Doğrulama	71,234 / 15	73,718 / 19	73,077 / 21



Şekil 3.9 α=0,3 İterasyonların Sonuçları

3.2.3.4 Önerilen Yöntemin $\alpha = 0,4$ Durumundaki Sonuçları

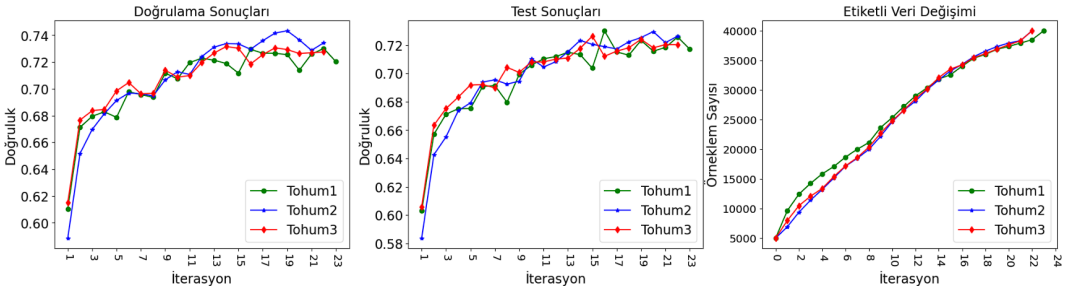
α'nın 0,4 alındığında elde edilen nihai sonuçlar Tablo 3.16'da, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.17'de verilmiştir. Şekil 3.10'da ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.16 α=0,4 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	71,71	71,71	72,01	71,68	7396,92
	72,63	72,59	73,04	72,59	6806,23
	72,03	72,03	72,11	71,90	6895,94
Doğrulama	72,03	72,23	72,14	71,95	7396,92
	73,44	73,19	73,42	73,15	6806,23
	72,75	72,55	72,65	72,44	6895,94

Tablo 3.17 α=0,4 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	73,017 / 16	72,957 / 20	72,636 / 15
Doğrulama	73,017 / 22	74,339 / 19	73,157 / 14



Şekil 3.10 α=0,4 İterasyonların Sonuçları

3.2.3.5 Önerilen Yöntemin $\alpha = 0,5$ Durumundaki Sonuçları

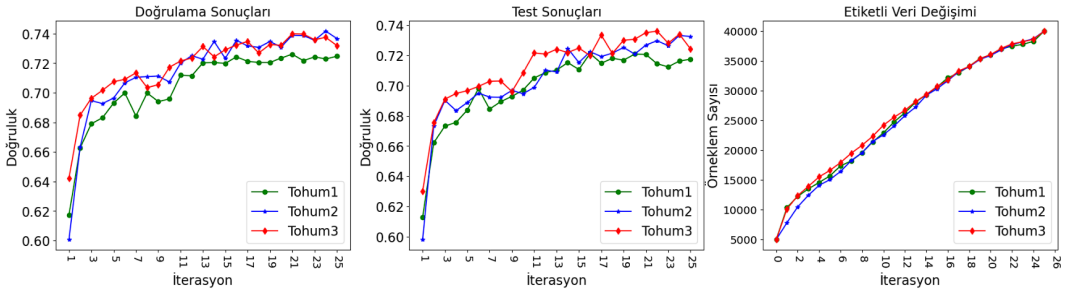
α'nın 0,5 alındığında elde edilen nihai sonuçlar Tablo 3.18'de, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.19'da verilmiştir. Şekil 3.11'de ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.18 α=0,5 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	71,75	71,41	72,08	71,55	7997,32
	73,25	73,57	73,69	73,29	7942,45
	72,43	72,27	72,43	72,25	7912,29
Doğrulama	72,47	72,08	72,54	72,15	7997,32
	73,67	73,63	73,63	73,29	7942,45
	73,19	72,79	73,07	72,86	7912,29

Tablo 3.19 α=0,5 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	72,135 / 16	73,337 / 24	73,618 / 22
Doğrulama	72,616 / 21	74,179 / 24	73,998 / 21



Şekil 3.11 α=0,5 İterasyonların Sonuçları

3.2.3.6 Önerilen Yöntemin $\alpha = 0,6$ Durumundaki Sonuçları

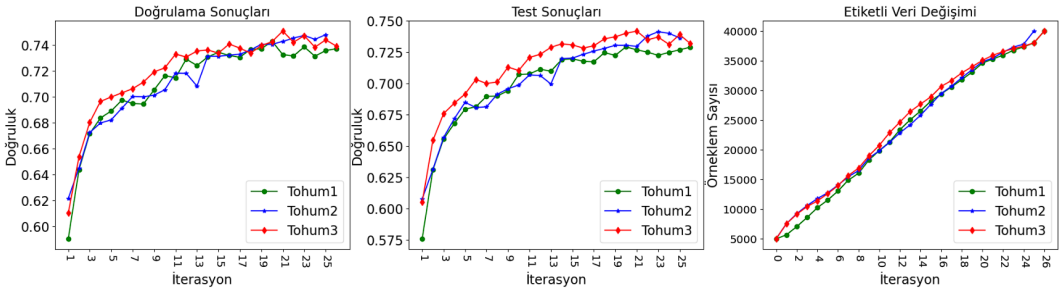
α'nın 0,6 alındığında elde edilen nihai sonuçlar Tablo 3.20'de, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.21'de verilmiştir. Şekil 3.12'de ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.20 α=0,6 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	72,89	72,71	73,23	72,62	8098,03
	73,64	73,59	74,03	73,63	7392,75
	73,18	72,94	73,21	72,89	7741,31
Doğrulama	73,72	73,66	73,75	73,35	8098,03
	74,79	74,61	74,79	74,55	7392,75
	73,92	73,57	73,79	73,49	7741,31

Tablo 3.21 α=0,6 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	72,917 / 20	74,119 / 23	74,179 / 20
Doğrulama	74,299 / 20	74,80 / 25	75,08 / 21



Şekil 3.12 α=0,6 İterasyonların Sonuçları

3.2.3.7 Önerilen Yöntemin $\alpha = 0,7$ Durumundaki Sonuçları

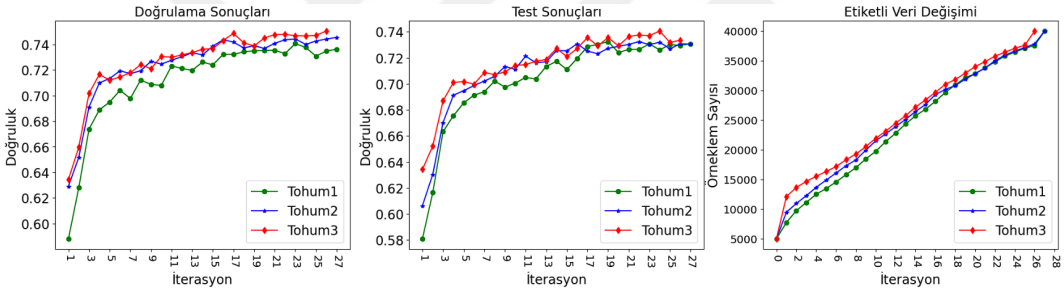
α'nın 0,7 alındığında elde edilen nihai sonuçlar Tablo 3.22'de, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.23'te verilmiştir. Şekil 3.13'te ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.22 α=0,7 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	73,07	72,72	73,41	72,84	8065,58
	73,06	73,61	73,43	73,25	8285,86
	73,36	73,60	73,42	73,41	7814,11
Doğrulama	73,62	73,29	73,67	73,32	8065,58
	74,54	74,75	74,49	74,37	8285,86
	75,04	75,05	74,93	74,93	7814,11

Tablo 3.23 α=0,7 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	73,237 / 19	73,237 / 22	74,058 / 24
Doğrulama	74,058 / 23	74,539 / 27	75,04 / 26



Şekil 3.13 α=0,7 İterasyonların Sonuçları

3.2.3.8 Önerilen Yöntemin $\alpha = 0,8$ Durumundaki Sonuçları

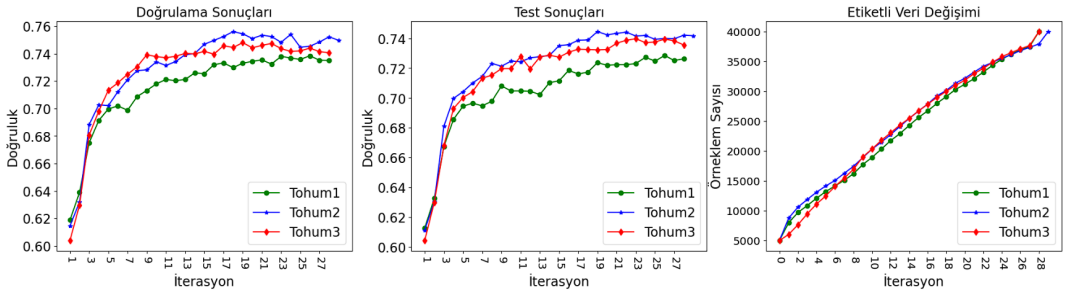
α'nın 0,8 alındığında elde edilen nihai sonuçlar Tablo 3.24'te, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.25'te verilmiştir. Şekil 3.14'te ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.24 α=0,8 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	72,64	72,46	72,92	72,57	8278,30
	74,18	74,31	74,53	74,35	8654,83
	73,56	73,46	73,61	73,43	8109,40
Doğrulama	73,49	73,39	73,60	73,43	8278,30
	74,96	74,79	74,93	74,81	8654,83
	74,06	73,79	73,94	73,73	8109,40

Tablo 3.25 α=0,8 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	72,857 / 26	74,459 / 19	73,978 / 23
Doğrulama	73,838 / 26	75,601 / 18	74,80 / 19



Şekil 3.14 α=0,8 İterasyonların Sonuçları

3.2.3.9 Önerilen Yöntemin $\alpha = 0,9$ Durumundaki Sonuçları

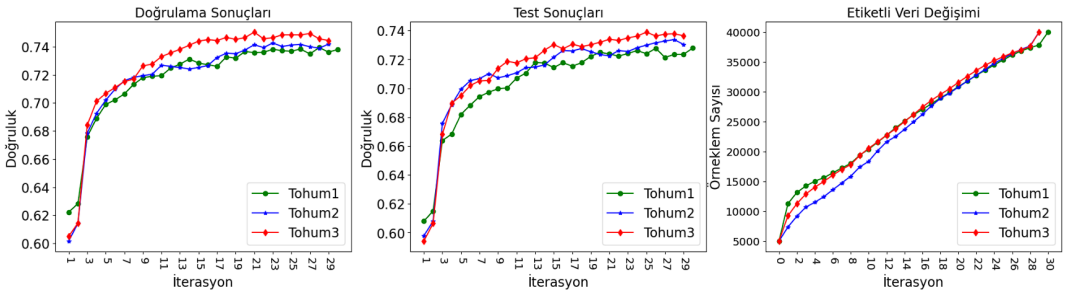
α'nın 0,9 alındığında elde edilen nihai sonuçlar Tablo 3.26'da, iterasyonlardaki en başarılı sonuçlar ise Tablo 3.27'de verilmiştir. Şekil 3.15'te ise her bir iterasyona ait doğruluk grafikleri ile etiketli verinin değişimi verilmiştir.

Tablo 3.26 α=0,9 Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	72,79	72,63	73,11	72,61	9336,41
	73,04	73,01	73,43	73,02	9119,62
	73,64	73,89	73,99	73,83	8869,90
Doğrulama	73,79	73,67	73,85	73,59	9336,41
	74,18	73,90	74,14	73,81	9119,62
	74,44	74,36	74,41	74,33	8869,90

Tablo 3.27 α=0,9 En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	72,796 / 30	73,377 / 28	73,898 / 25
Doğrulama	73,938 / 28	74,259 / 23	75,04 / 21



Şekil 3.15 α=0,9 İterasyonların Sonuçları

3.2.4 Adaptif α Yaklaşımı

Elde edilen sonuçlar incelendiğinde ilk iterasyonda tekil modelin başarımı sürekli artış gösterirken, ilerleyen iterasyonlarda belli bir doygunluğa erişmiştir. Doygunluk seviyesinde ise hafif dalgalanmaların olduğu görülmektedir. Bu durum göz önüne alındığında sabit bir α yaklaşımı ne kadar doğru olur? Bir başka deyişle ilk iterasyonlarda ϕ_M sürekli artış halinde olduğundan ϕ_E'ye göre daha ağırlıklı olması gerekmez mi? ϕ_E'nin daha başarılı olduğu durumlarda da ϕ_M'e göre daha ağırlıklı olması gerekmez mi? Ayrıca ϕ_E ve ϕ_M aynı doğruluk değerlerine sahip olduğunda ise α'nın 0,5 yani ϕ_M ve ϕ_E'nin ağırlıksız ortalamasınının hesaplanması gerekli değil midir? Bu sebeplerden dolayı sabit bir α kullanmak yerine ϕ_M ve ϕ_E'nin sahip oldukları başarımlar doğrultusunda adaptif bir α kullanılması daha doğru olacaktır. Adaptif α'nın üretimi Denklem (3.1) kullanılarak sağlanmıştır.

$$\alpha = 0,5 + ((ACC_E/ACC_M)^2 - 1) * 4 \tag{3.1}$$

ACC_E ve ACC_M sırasıyla ϕ_E ve ϕ_M'nin doğrulama verisi üzerindeki doğruluk değerleridir. ACC_M değeri, hali hazırda *obtain-current-probabilities* fonksiyonu içerisinde hesaplanan V_{MR}[-1] değerine karşılık gelmektedir.

Formülün incelenmesi:

- ACC_E ve ACC_M birbirlerine denk olduğunda, ϕ_M ve ϕ_E'nin ağırlıksız ortalaması yani α'nın 0,5 olması gerekir. Denklem (3.1)'deki $(ACC_E/ACC_M)^2 - 1)$ kısmı bu durumda 0 üretecektir ve nihai α değeri 0,5 olacaktır.
- ACC_E'nin ACC_M'den büyük olduğu durumlarda, ϕ_E'nin daha baskın olmasını gerekmektedir ve Denklem (3.1)'deki $(ACC_E/ACC_M)^2 - 1)$ değeri

pozitif sonuç verecektir. Örneğin: $(0,72/0,70)^2 - 1$'in sonucu 0,0579 gelecektir ve bu sayı 4 ile çarpılıp, 0,5 eklendiğinde nihai α değeri yaklaşık olarak 0,732 olacaktır.

- ACC_M'nin ACC_E'den büyük olduğu durumlarda, ϕ_M'nin daha baskın olmasını gerekmektedir ve Denklem (3.1)'deki $(ACC_E/ACC_M)^2 - 1)$ değeri negatif olacaktır. Örneğin: $(0,70/0,72)^2 - 1$'in sonucu -0,0547 gelecektir ve bu sayı 4 ile çarpılıp, 0,5 eklendiğinde nihai α değeri yaklaşık olarak 0,281 olacaktır.

- Hesaplanan α değerinin 0,05 - 0,95 arasında kalmasını sağlamak için kontrol edilmelidir. Bu kontrol find-alpha fonksiyonu içerisindeki if-elseif-else ile sağlanmıştır.

- Çarpım teriminin 4 seçilmesindeki sebep ise ACC_M ve ACC_E arası %5'lik bir fark oluşması durumunda α değerinin yaklaşık olarak 0,9 ve 0,1 gelmesidir.

- Çarpım terimi 4 yerine büyük değerler alınırsa (Örneğin: 10, 15 gibi) eğer nihai α değeri sürekli olarak 0,95 veya 0,05 olacaktır. Küçük değerler alınırsa ise (Örneğin: 0,5, 1 gibi) nihai α değeri 0,4 - 0,6 arasına sıkışıp kalacaktır.

obtain-ensemble-probabilities fonksiyonunun güncellenmesi: α'nın artık girdi olarak verilmemesi gereklidir. Her *nto* $\leftarrow$ *(1-*α*) * ato +* α ** kto* satırından sonra $\alpha \leftarrow$ *find-alpha(kdo)* çalıştırılmalıdır. *find-alpha* fonksiyonu aşağıdaki gibidir.

Algorithm 8 find-alpha

Girdiler: Doğrulama verisindeki kaydedilmiş olasılıkların kolektif sonucu(kdo)
Çıktılar: Alpha(α)

global V_{MR};
$ACC_E \leftarrow$ kdo'nun doğruluk değeri;
$\alpha \leftarrow 0,5 + ((ACC_E/V_{\text{MR}}[-1])^2 - 1) * 4$;
if $\alpha > 0,95$ **then**
  $\alpha \leftarrow 0,95$;
  return α;
else if $\alpha < 0,05$ **then**
  $\alpha \leftarrow 0,05$;
  return α;
else
  return α;
end

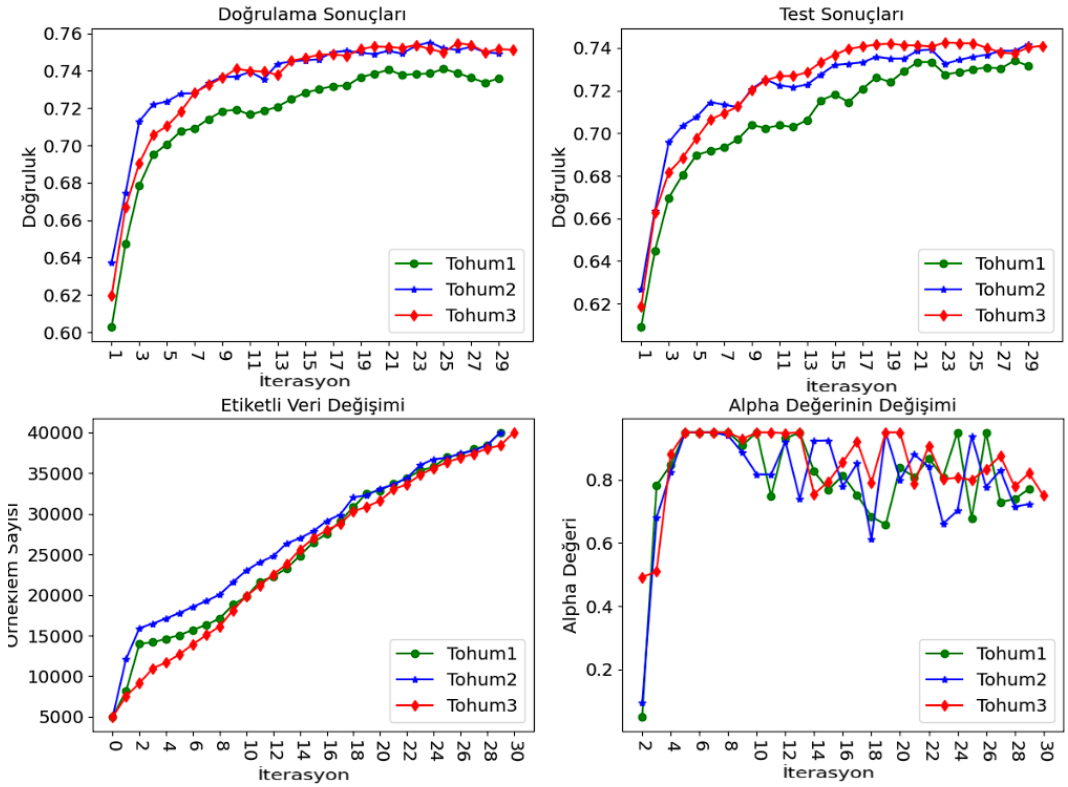
Önerilen adaptif α yaklaşımı CIFAR10 veri seti ile DenseNet121 mimarisinde denenmiştir. Her iterasyonda elde edilen sonuçlar Şekil 3.16'da, nihai sonuçlar Tablo 3.28'de ve en iyi andaki sonuçlar Tablo 3.29'da verilmiştir.

Tablo 3.28 Adaptif α Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	73,16	72,94	73,45	72,99	9020,80
	74,19	74,26	74,52	74,23	9142,07
	74,10	74,13	74,13	73,99	8823,01
Doğrulama	73,60	73,43	73,66	73,43	9020,80
	74,94	74,84	74,86	74,69	9142,07
	75,12	74,92	74,98	74,86	8823,01

Tablo 3.29 Adaptif α En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	73,397 / 28	74,179 / 29	74,259 / 23
Doğrulama	74,099 / 25	75,521 / 24	75,481 / 26



Şekil 3.16 Adaptif α İterasyonların Sonuçları

Şekil 3.16'deki alpha değerinin değişimi incelendiğinde: ikinci iterasyon sonucunda α'nın 0,5'den düşük olduğu, sonraki bir kaç iterasyonda ise sürekli 0,95 olduğu görülmüştür. İlerleyen iterasyonlar ile α 0,8 civarında değerler almıştır.

3.2.5 Farklı α Değerlerinin Karşılaştırılması

α değerinin önerilen yöntem üzerindeki etkisinin sonuçları bu kısımda incelenmiştir. Verilen görseller ve tablodaki değerler tohumların ortalama değerleri üzerinden hesaplanmıştır.

Son iterasyondaki nihai sonuçlar ile en iyi iterasyondaki doğruluk değerlerinin, var olan yönteme göre ortalama başarımları Tablo 3.30'da verilmiştir. Nihai sonuçlar Tablo 3.4'ün DenseNet121 sonuçları ile en iyi iterasyon ise Tablo 3.7'nin DenseNet121 sonuçlarıyla kıyaslanmıştır.

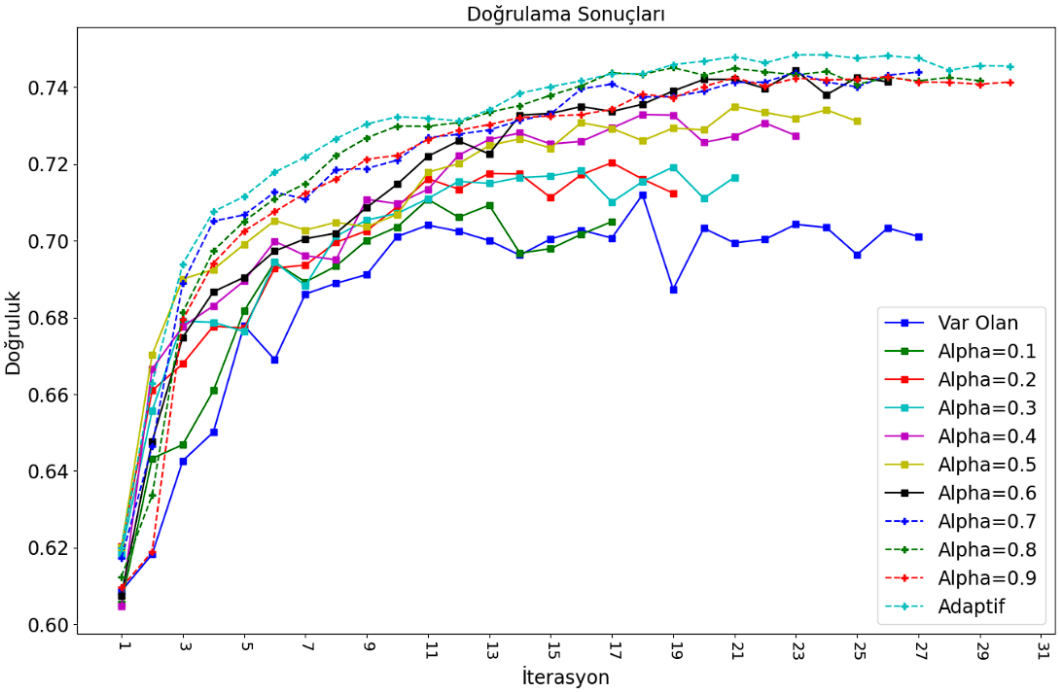
Tablo 3.30 CIFAR-10 Var Olan Yönteme Göre Yüzdelik Başarımları

		Nihai Sonuçlar				En İyi İterasyon
Veri	α Değeri	Doğruluk	Kesinlik	Duyarlılık	F1	Doğruluk
Test	0,1	0,433	0,214	0,457	0,245	-0,075
	0,2	1,847	1,510	1,818	1,758	1,956
	0,3	2,211	2,080	2,221	2,192	1,692
	0,4	3,836	3,408	3,812	3,876	3,188
	0,5	4,346	3,854	4,318	4,317	3,413
	0,6	5,432	4,803	5,394	5,303	4,416
	0,7	5,326	5,133	5,298	5,479	4,093
	0,8	5,749	5,275	5,678	5,889	4,453
	0,9	5,316	4,942	5,427	5,461	3,876
	Adaptif	6,258	5,795	6,175	6,305	4,710
Doğrulama	0,1	0,553	0,320	0,542	0,356	-0,187
	0,2	1,610	1,549	1,634	1,655	1,214
	0,3	2,201	1,958	2,164	2,087	1,710
	0,4	3,752	3,463	3,789	3,812	2,868
	0,5	4,286	3,722	4,282	4,167	2,999
	0,6	5,752	5,303	5,743	5,655	4,579
	0,7	6,114	5,897	6,112	6,241	4,326
	0,8	5,791	5,364	5,813	5,933	4,607
	0,9	5,744	5,347	5,783	5,811	4,140
	Adaptif	6,334	5,943	6,307	6,405	5,009

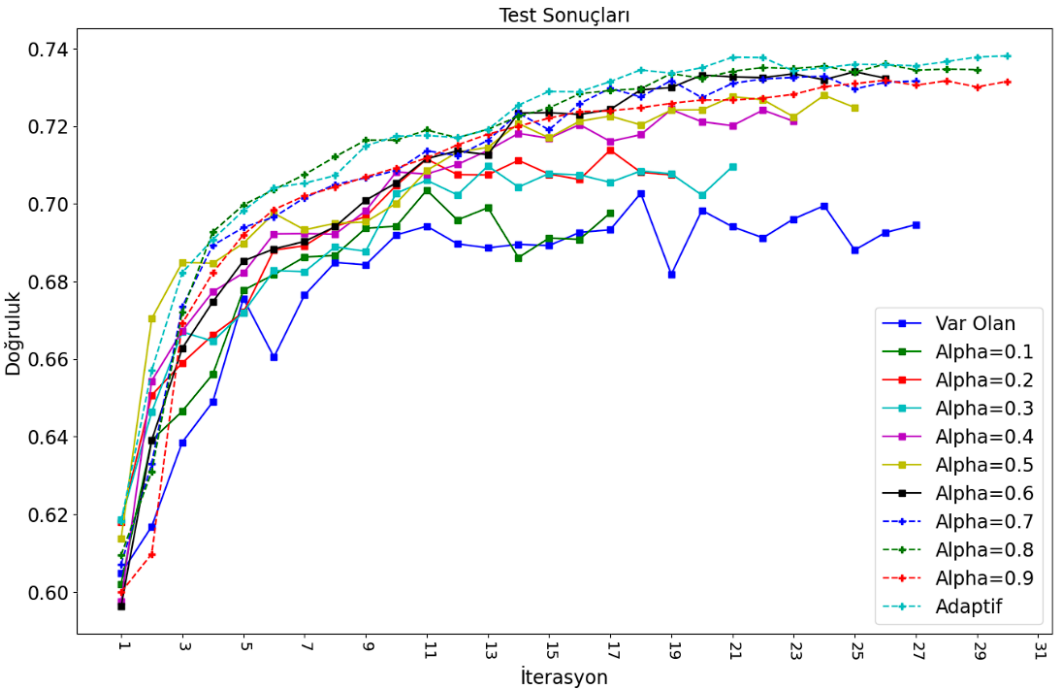
Tablo 3.30'dan da anlaşılacağı üzere önerilen yöntem daha başarılı sonuçlar vermiştir. Adaptif α yaklaşımı hem test hem de doğrulama verisi üzerindeki en iyi sonucu vermiştir. Sabit α değerlerinde: nihai sonuçlarda doğrulama ve test verilerinin en iyi sonuçları sırasıyla 0,7 ve 0,8 değerlerinde elde edilmişken, en iyi

iterasyon sonuçlarında ise doğrulama ve test verilerinin en yüksek başarımı 0,8'de elde edilmiştir.

Var olan yöntemin ve bütün α değerlerinin ortalama doğruluk grafikleri, doğrulama verisi için Şekil 3.17'de, test verisi için ise Şekil 3.18'de verilmiştir.



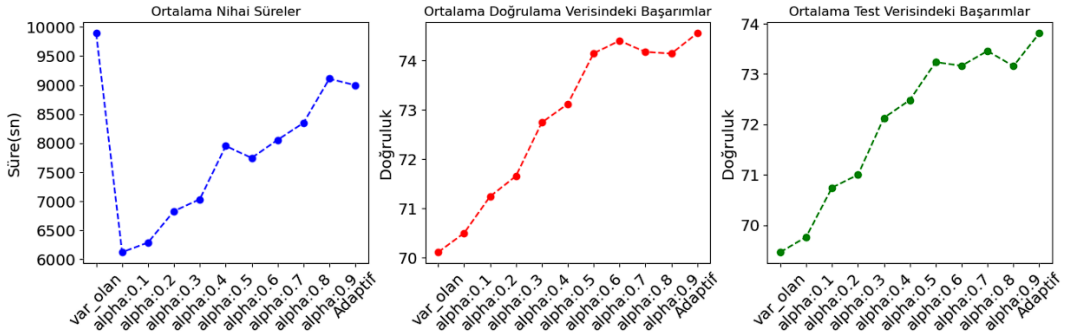
Şekil 3.17 Doğrulama Verisindeki Ortalama Başarımlar



Şekil 3.18 Test Verisindeki Ortalama Başarımlar

Şekil 3.17 ve 3.18 incelendiğinde önerilen yöntemin bariz bir şekilde var olan yönteme göre başarılı olduğu görülmüştür. Var olan yöntemin doyum noktası 0,7 etrafında iken, önerilen yöntem α değerine bağlı olarak 0,74'e kadar ulaşmayı başarmıştır. α'nın 0,6, 0,7, 0,8 ve 0,9 değerleri en iyi sonuçları vermiştir. Ayrıca var olan yöntem sadece anlık eğitilen modele bağlı olduğunda başarımında kırılımlar görülmüştür. Önerilen yöntemde ise α değeri arttıkça bu kırılımlar yerini hafif dalgalanmalara bırakmıştır. Bunun sebebi eski modellerin ortaklaşa karar almasıdır. Adaptif α yaklaşımı ise durumu biraz daha iyileştirmiştir.

Nihai sonuçların elde edilmesinde geçen ortalama süreler ve başarımları Şekil 3.19'da verilmiştir.



Şekil 3.19 Ortalama Nihai Süre ve Başarımların Değişimleri

Şekil 3.19 incelendiğinde, α değerinin 0,5'den 0,6'a geçince sürede görülen azalmanın sebebi: iteratif akışı sonlandırmak için gerekli olan sözde etiket miktarının 0,95*N'den 0,925*N'e düşürülmüş olmasıdır. Bunun haricinde α değerinin sürekli artışı ile birlikte ihtiyaç duyulan süre artış göstermiştir. Adaptif α yaklaşımı ise 0,9 ile hemen hemen aynı sürede sonuç üretmesine rağmen daha yüksek başarım elde etmiştir.

Çalışmanın kalan kısmında önerilen yöntemin sonuçları elde edilirken en yüksek sonucu veren adaptif α yaklaşımı kullanılmıştır.

3.2.6 Farklı Mimari Bağımsızlığı

CIFAR10 veri kümesi ile EfficientNetB0 mimarisinin adaptif α yaklaşımı ile elde edilen nihai sonuçları Tablo 3.31'de, en iyi iterasyondaki sonuçları ise Tablo 3.32'de verilmiştir.

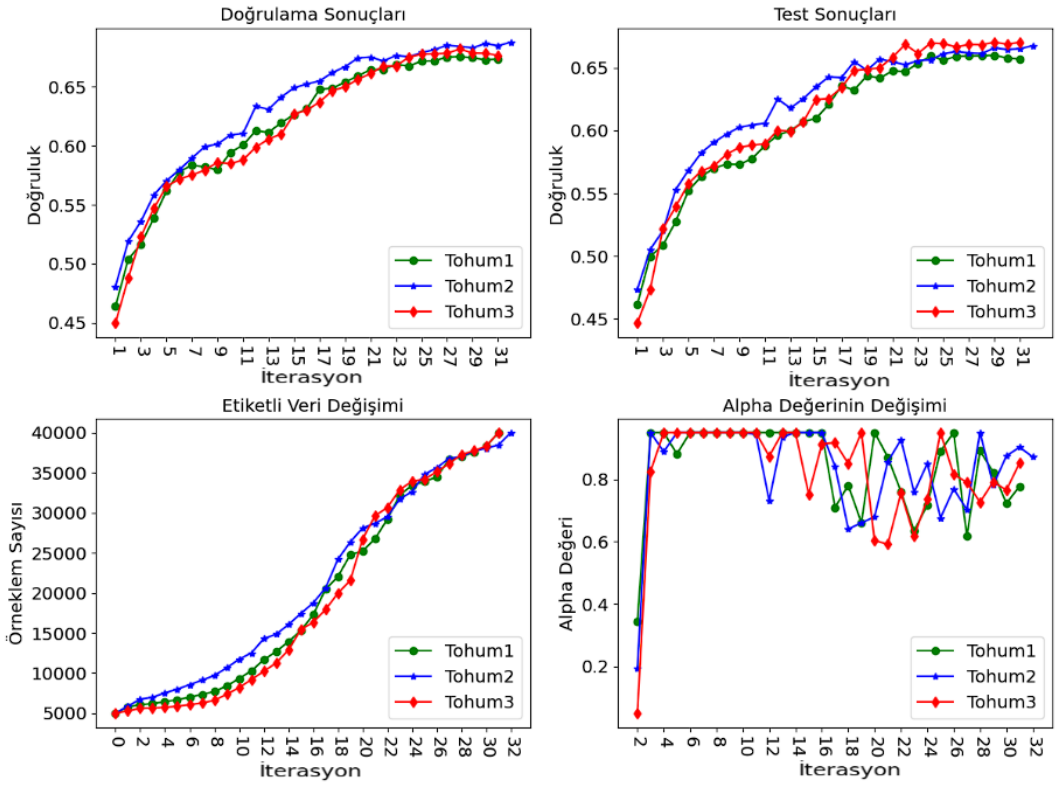
Tablo 3.31 Adaptif α Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
Test	65,68	65,24	66,12	65,13	9676,11
	66,75	66,57	67,20	66,79	10114,37
	67,03	66,81	67,02	66,48	9577,80
Doğrulama	67,35	66,89	67,39	66,78	9676,11
	68,77	68,25	68,70	68,38	10114,37
	67,65	67,55	67,45	67,02	9577,80

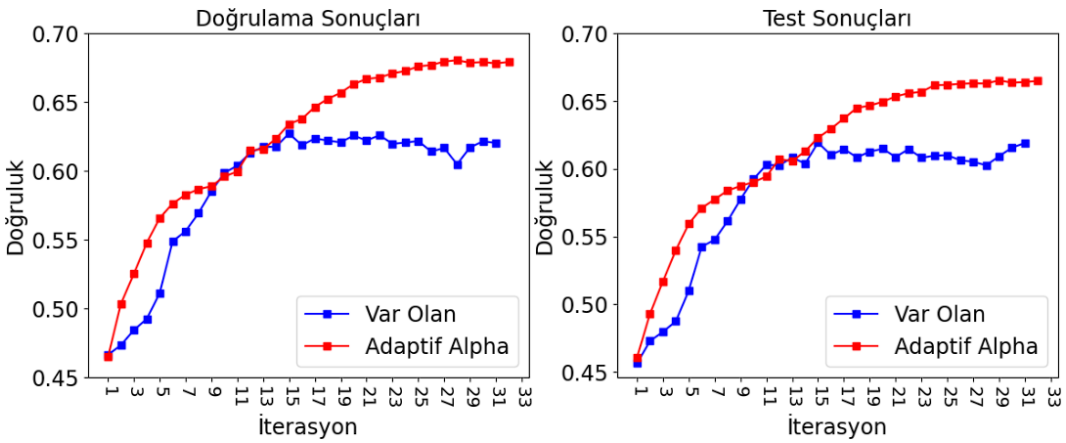
Tablo 3.32 Adaptif α En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	65,986 / 29	66,747 / 32	67,027 / 29
Doğrulama	67,548 / 28	68,77 / 32	68,189 / 28

Önerilen yöntemin her bir iterasyondaki sonuçları Şekil 3.20'de, var olan yönteme göre ortalama sonuçlarının grafikleri Şekil 3.21'de verilmiştir.



Şekil 3.20 Adaptif α İterasyonların Sonuçları



Şekil 3.21 Test ve Doğrulama Verilerindeki Ortalama Başarımlar

Elde edilen sonuçlara göre adaptif α yaklaşımı CIFAR10 ve EfficientNetB0 mimarisi altındaki ortalama sonuçlar:

- Tablo 3.4'teki EfficientNetB0 kısmındaki ortalama nihai sonuçlara ve başarım metriklerine göre test verisinde %7,41, %6,73, %7,36, %8,88, doğrulama verisinde ise %9,48, %9,03, %9,45, %11,33 daha başarılıdır.
- Tablo 3.7'deki EfficientNetB0 kısmındaki ortalama en iyi iterasyon sonuçlarına ve başarım metriğine göre test verisinde %5,96, doğrulama verisinde %7,88 daha yüksek başarıya sahiptir.

Şekil 3.21 incelendiğinde: onbeşinci iterasyonda hem önerilen yöntemin hem de var olan yöntem aynı seviyelerde olduğu görülmüştür. Ancak onbeşinci iterasyonda var olan yöntemin ortalama 33.000 etiketli örnekleme sahipken, önerilen yöntem 16.000 etiketli örnekleme sahiptir. Bu durumdan dolayı var olan onbeşinci iterasyonda doygunluğa erişmişken, önerilen yöntem ilerleyen iterasyonlarda başarımını arttırabilmiştir.

EfficientNetB0 mimarisinde de önerilen yöntem daha iyi sonuç verdiği için önerilen yöntemin mimariden bağımsızlığı kanıtlanmıştır.

3.2.7 Intel Veri Kümesi Deneysel Sonuçlar

Önerilen yöntemin veri kümesinden bağımsızlığını göstermek için Intel veri kümesi kullanılarak deneysel çalışmalar yapılmıştır. Sonuçlar adaptif α yaklaşımıyla elde edilmiştir.

Tablo 3.33'te önerilen yöntemin, iterasyonlar sonucunda elde edilmiş olan nihai sonuçlara yer verilmiştir.

Tablo 3.33 Intel Nihai Sonuçları

Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
EfficientNetB0	Test	80,72	80,84	80,97	80,70	6527,17
		82,31	82,59	82,47	82,41	6455,68
		81,01	81,67	81,32	81,25	6659,89
	Doğrulama	82,78	82,86	82,93	82,77	6527,17
		83,49	83,89	83,63	83,57	6455,68
		80,48	80,91	80,78	80,59	6659,89
DenseNet121	Test	84,79	84,80	85,07	84,87	7321,26
		85,44	85,65	85,64	85,59	8068,43
		83,78	84,00	84,04	83,99	6256,39
	Doğrulama	86,26	86,28	86,41	86,31	7321,26
		85,73	85,99	85,95	85,87	8068,43
		84,85	85,12	85,02	84,96	6256,39

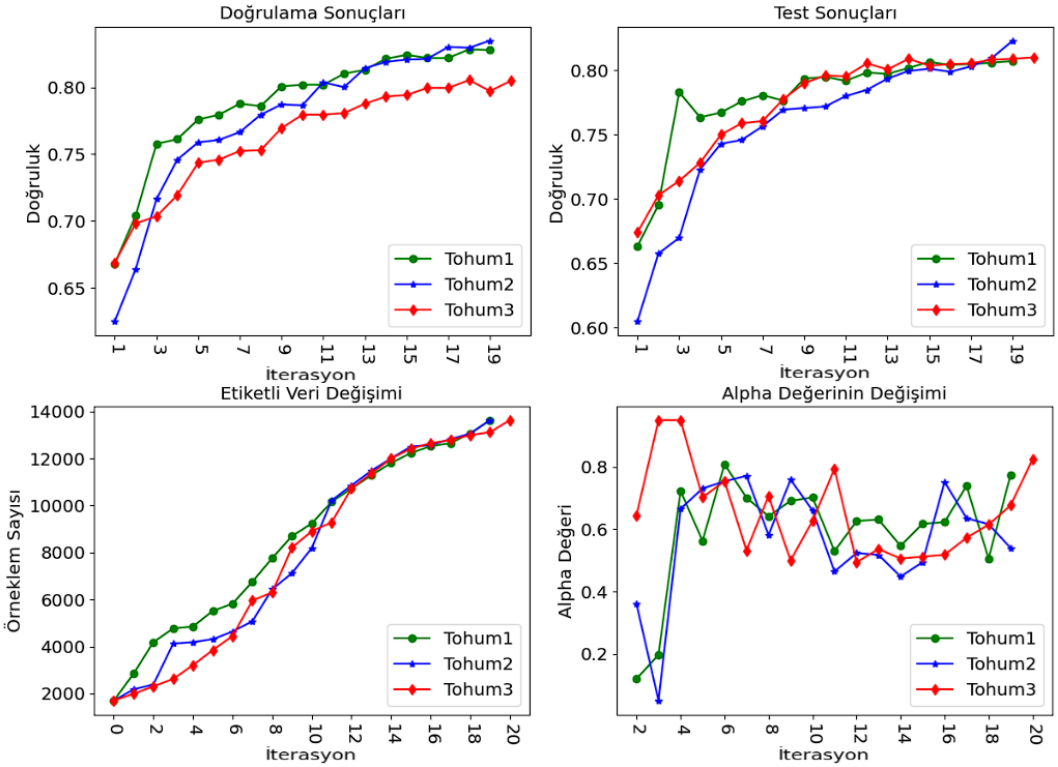
Tablo 3,34'te ise önerilen yöntemin, iterasyonlar esnasında en iyi başarıma sahip iterasyonun sonucu verilmiştir,

Tablo 3.34 Intel En Başarılı Andaki Doğruluk Sonuçlar

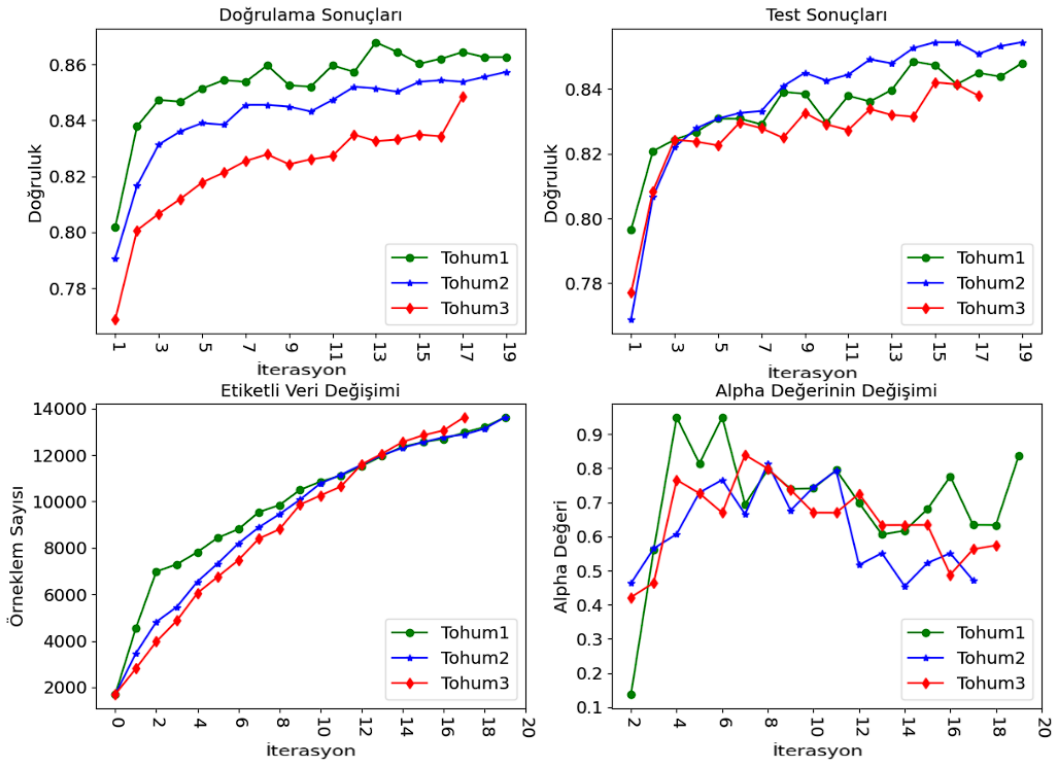
Mamari	Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
EfficientNetB0	Test	80,719 / 19	82,311 / 19	81,014 / 20
	Doğrulama	82,842 / 18	83,491 / 19	80,542 / 18
DenseNet121	Test	84,847 / 14	85,436 / 19	84,198 / 15
	Doğrulama	86,792 / 13	85,731 / 19	84,847 / 17

Şekil 3.22 ve Şekil 3.23'te sırasıyla EfficientNetB0 ve DenseNet121 mimarilerinin her iterasyondaki doğrulama ve test verileri üzerindeki doğruluk değerleri, etiketli veri miktarları ve α değerleri grafik olarak verilmiştir.

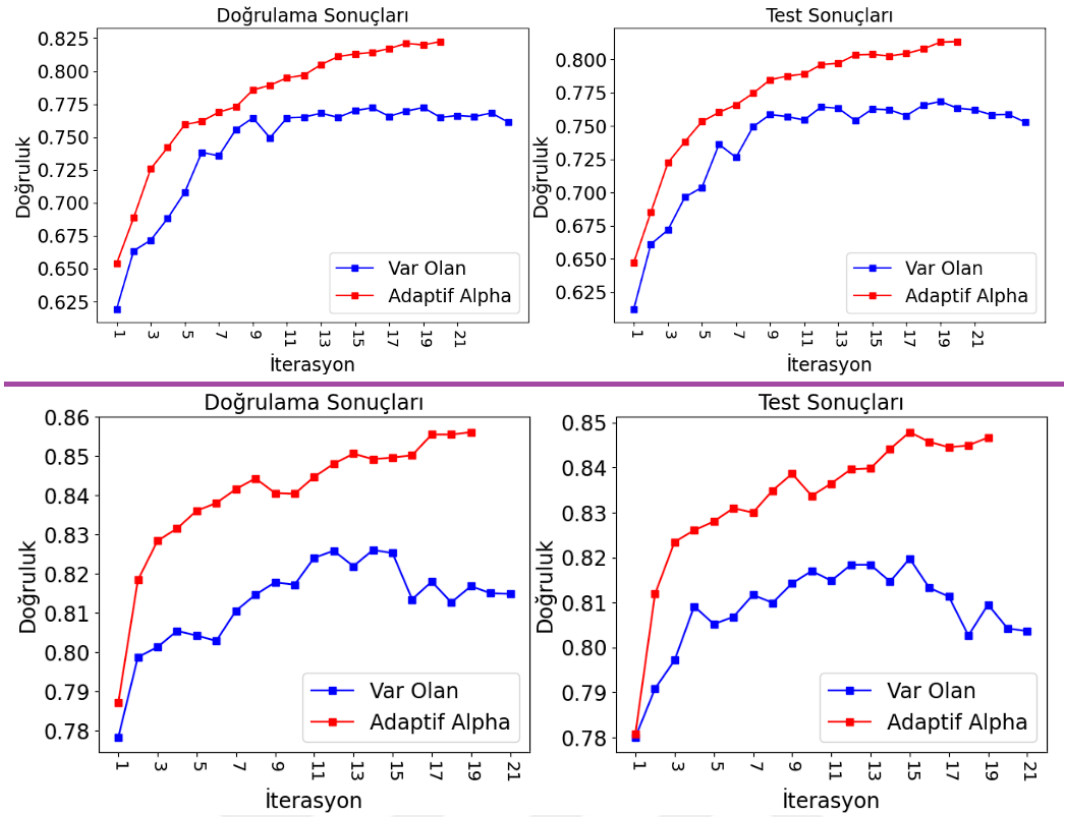
Şekil 3.24'te ise var olan yönteme göre her iterasyondaki ortalama değerlerin grafikleri verilmiştir.



Şekil 3.22 EfficientNetB0 İterasyonların Sonuçları



Şekil 3.23 DenseNet121 İterasyonların Sonuçları



Şekil 3.24 Var olan yöntem ile karşılaştırmalı iterasyonların ortalama sonuçları. Üst satır EfficientNetB0, alt satır DenseNet121 mimarilerine aittir.

Tablo 3.35'te önerilen yöntemin, var olan yönteme göre yüzdelik başarımları verilmiştir.

Tablo 3.35 Intel Var Olan Yönteme Göre Yüzdelik Başarımları

		Nihai Sonuçlar				En İyi İterasyon
Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Doğruluk
EfficientNetB0	Test	8,07	8,50	8,12	8,19	5,16
	Doğrulama	8,05	8,47	8,05	8,17	5,41
DenseNet121	Test	5,35	5,32	5,33	5,40	3,58
	Doğrulama	5,06	5,15	5,10	5,14	2,76

Tablo 3.35'ten anlaşılacağı üzere veri kümesi aynı kalsa bile mimarinin farklı olması yüzdelik başarımları etkilemiştir. Buna rağmen önerilen yöntem daha yüksek başarıma sahiptir.

3.2.8 Gtzan Veri Kümesi Deneysel Sonuçlar

Önerilen yöntemin Gtzan veri kümesi ve ResNet34 mimarisi kullanılarak elde edilen sonuçlar bu bölümde verilmiştir. Sonuçlar adaptif α yaklaşımıyla elde edilmiştir.

Tablo 3.36'da önerilen yöntemin, iterasyonlar sonucunda elde edilmiş olan nihai sonuçlara yer verilmiştir.

Tablo 3.36 GTZAN Nihai Sonuçları

Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre
ResNet34	Test	63,50	63,29	62,46	60,67	4468,64
		64,00	68,28	62,80	59,62	4794,47
		64,25	63,29	62,83	61,03	4655,22
	Doğrulama	65,00	64,23	64,79	63,42	4468,64
		66,00	60,37	65,75	62,25	4794,47
		67,75	70,52	67,05	65,58	4655,22

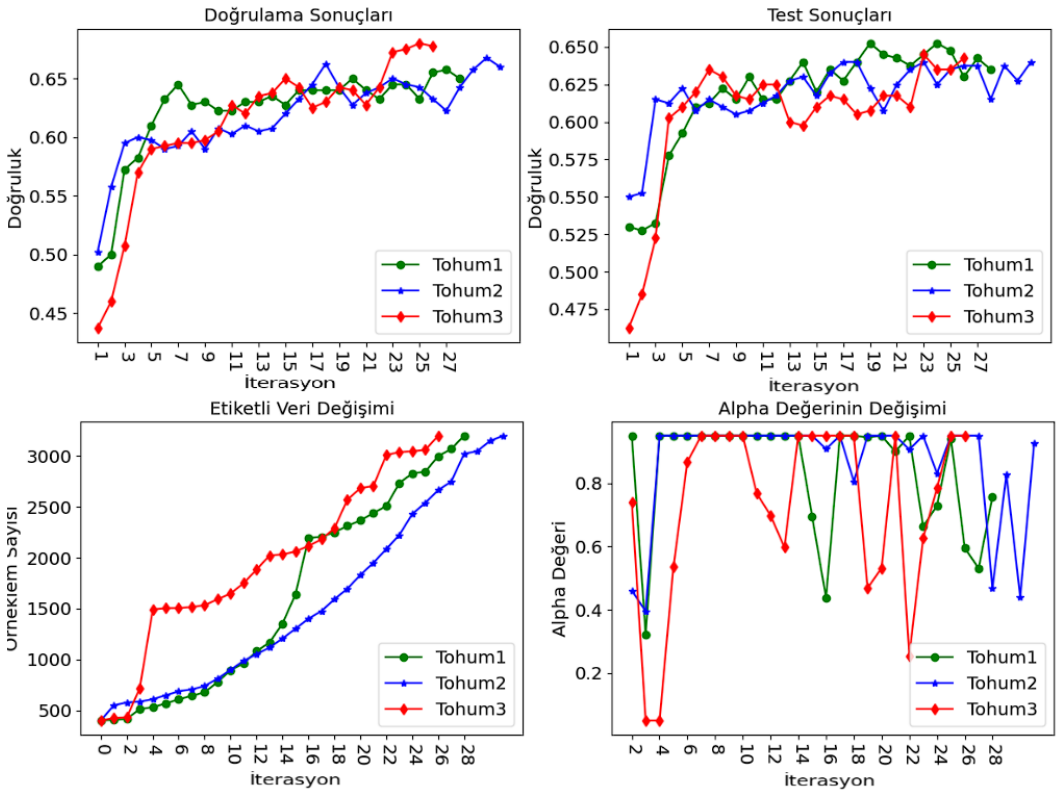
Tablo 3.37'de ise önerilen yöntemin, iterasyonlar esnasında en iyi başarıma sahip iterasyonun sonucu verilmiştir.

Tablo 3.37 GTZAN En Başarılı Andaki Doğruluk Sonuçlar

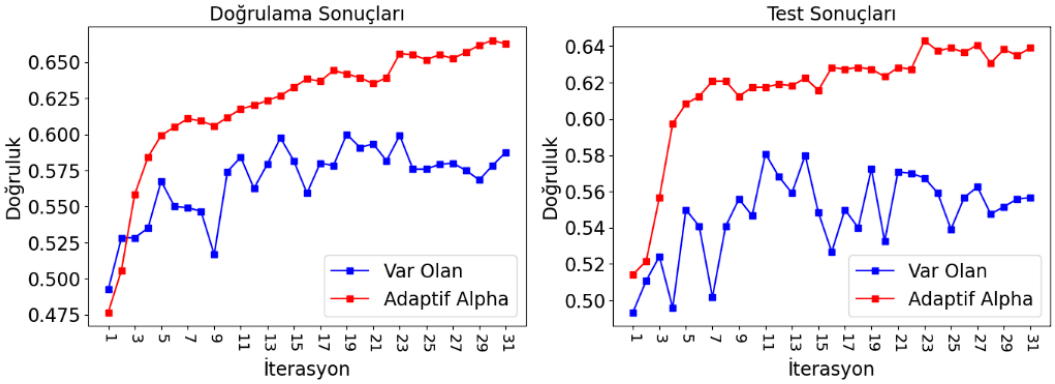
Mamari	Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
ResNet34	Test	65,25 / 24	64,00 / 31	64,50 / 23
	Doğrulama	65,75 / 27	66,75 / 30	68,00 / 25

Şekil 3.25'te ResNet34 mimarisinin her iterasyondaki doğrulama ve test verileri üzerindeki doğruluk değerleri, etikeli veri miktarları ve α değerleri grafik olarak verilmiştir.

Şekil 3.26'da ise var olan yönteme göre her iterasyondaki ortalama değerlerin grafikleri verilmiştir.



Şekil 3.25 ResNet34 İterasyonların Sonuçları



Şekil 3.26 Var olan yöntem ile karşılaştırmalı iterasyonların ortalama sonuçları.

Tablo 3.38'de önerilen yöntemin, var olan yönteme göre yüzdelik başarımları verilmiştir.

Tablo 3.38 GTZAN Var Olan Yönteme Göre Yüzdelik Başarımları

		Nihai Sonuçlar				En İyi İterasyon
Mimari	Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Doğruluk
ResNet34	Test	14,82	14,50	14,19	13,71	8,81
	Doğrulama	12,76	12,15	14,03	13,51	6,45

Elde edilen tüm deneysel sonuçlar doğrultusunda, geçmiş iterasyonlardaki modellerin oluşturduğu kolektif sonuç hesaplanırken sürekli olarak son yedi model kullanılmıştır. İlk iterasyonlardaki modeller daha az veri ile eğitildiğinden, başarımları görece daha düşük olup, bu modellerin ilerleyen iterasyonlarda kullanılması nihai sonucu olumsuz yönde etkileyebilmektedir. Geçmiş iterasyonlardaki modellerin oluşturduğu kolektif sonuç ile anlık modelin sonucunun birleştirilmesinde kullanılan α katsayısı nihai sonucu doğrudan etkilemiştir. Ayrıca, önerilen yöntemin sonuç üretme süresi daha uzun olduğundan, bu durum daha fazla risk alınmasını gerektirmiştir.

4
KSIITLARIN İNCELENMESİ

Bu bölümde, önerilen ve var olan yöntemlerinin sonuçları elde edilirken kullanılan kısıtlar incelenmiştir. Verilen deneysel sonuçlar CIFAR10 veri seti ile DenseNet121 mimarisi kullanılarak elde edilmiştir.

4.1 Olasılık Şartında Neden Azalma Yapılıyor ?

Sözde etiket üretiminde en güvenilir örneklemlerin seçilmesi gereklidir. Aksi takdirde yapılan yanlış etiketleme, sonraki iterasyonları da olumsuz etkileyecektir. Güvenilir örneklemlerin seçilebilmesi için olasılık şartı tanımlanır. Yapılan deneysel çalışmalarda olasılık şartı çok yüksek bir değerden başlatılarak, her iterasyonda belli miktar düşüş yapılmıştır. Bunun temel sebebi üretilen sözde etiket sayısıdır. Olasılık şartında düşüş yapılmadığında, üçüncü ve sonraki iterasyonlarda çok az sayıda örneklemin sözde etiketleri üretilebilmiştir. Bu durumda akışın sonlanması için çok daha fazla iterasyona ihtiyaç duyulacak olup, bu sebeple olasılık şartında düşüş yapılmıştır. Var olan yöntemde bu düşüşler daha az alınmıştır. Çünkü ani düşüş sonucu oluşabilecek yanlış etiketleme geriye kalan iterasyonları da olumsuz etkileyecektir. Önerilen yöntemde ise kolektif sonuç kullanıldığından bu durum baskılanabilmiştir.

4.2 Akış Neden Yüzdelik Olarak Sonlandırılıyor ?

Veri setleri içerisinde yer alan gürültülü örneklemlerin etiketlenmesi oldukça güçtür. Çünkü bu örneklemler birden fazla sınıfa benzerlik gösterebilir. Bir başka deyişle modelin gürültülü örneklemler üzerindeki sınıf olasılıkları birbirine çok yakın değerlerde olabilir. Örneğin: CIFAR10 veri setindeki gürültülü örneklemin sınıf olasılıklar [0,1, 0,19, 0,21, 0,22, 0,18, 0,1, 0, 0, 0, 0] olabilir ve bu durumda hiçbir zaman olasılık şartı sağlanmaz. Dolayısıyla sözde etiket miktarı istenilen miktara erişildiğinde, olasılık şartına bakılmaksızın kalan örneklemlerin sözde etiketleri üretilir. Gürültülü örneklemlerin etiketli veriye eklenmesi ile son iterasyonda

eğitilen modelin başarımında düşüşler görülebilir. Önerilen yöntemde istenilen miktar daha az seçilmiştir, bunun temel sebebi var olan yönteme göre süreyi fazla aşmamaktır. Var olan yöntemde ise daha düşük seçilmesinin sebebi son iterasyonda eğitilen modelin başarımını düşürmemektir. Çünkü son etiketleme işleminde hatalı etiketlenme ihtimali fazladır ve bu durum doğrudan başarıma etki eder. Önerilen yöntem de ise bu durum kolektiflik ile aşılmıştır.

4.3 Önerilen Yöntemde Neden Son Yedi Olasılık Kullanılıyor ?

ϕ_E değeri hesaplanırken dokuzuncu ve sonraki iterasyonlarda en son eklenen yedi olasılık kullanılır. Bu yöntemin tercih edilmesindeki sebep: ilk iterasyonlardaki güvenli etiketleme işlemiyle (olasılık şartı yüksek olduğundan en güvenilirler etiketlenir), eğitilen modelin başarımındaki sürekli artıştır. Bir başka deyişle doygunluk öncesinde model başarımı sürekli artış gösterdiğinden, ilerleyen iterasyonlarda ilk modelleri kullanmak doğru olmaz. Çünkü başarım olarak son yedi modelin olasılıkları her zaman ilk modellerin olasılıklarından daha yüksektir. Doygunluk seviyesinden sonra oluşan dalgalanmaların (eğitilen modelin başarımındaki dalgalanmalar) ϕ_E üzerindeki etkisini en aza indirgemek için modellerin etiketsiz verideki olasılıkları, doğrulama verisindeki doğruluk değerleriyle çarpılmıştır. Böylelikle daha başarılı modeller daha baskın olurken, nispeten daha başırısız modellerin etkisi daha az olur.

4.4 Süre Faktörünün İncelenmesi

Yapılan deneysel çalışmalarda temel amaç: var olan yönteme kıyasla daha iyi bir sonucu daha kısa veya yakın sürelerde elde etmek olduğundan önerilen yöntemde her iterasyon ile alınan risk daha da fazladır. Peki önerilen yöntem ile var olan yöntem, aynı kısıtlar altında çalıştırılırsa ne olur? Bu soruyu incelemek için iki farklı durum ele alınmıştır:

- Birinci Durum: Önerilen yöntemin, var olan yöntemdeki kısıtlar ile çalıştırılması.
- İkinci Durum: Var olan yöntemin, önerilen yöntemdeki kısıtlar ile çalıştırılması.

4.4.1 Birinci Durum

Önerilen yöntem Tablo 2.3'deki var olan yöntemin kısıtları altında çalıştırıldığında elde edilen sonuçlar Tablo 4.1, Tablo 4.2 ve Şekil 4.1'de verilmiştir. Sonuçlar elde

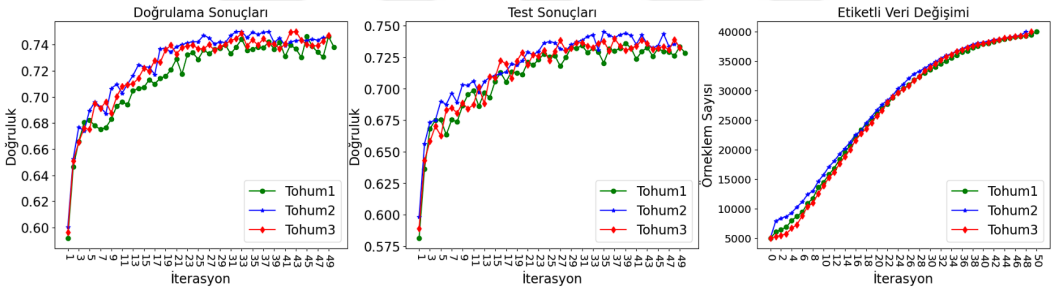
edilirken, ϕ_E ve ϕ_M'in ağırlıksız ortalaması olması için α 0.5 alınmıştır.

Tablo 4.1 Birinci Durum Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre(sn)
Test	72,84	72,62	73,16	72,69	16389,63
	73,54	73,83	73,91	73,68	16434,33
	73,28	73,24	73,34	73,01	16487,44
Doğrulama	73,82	73,61	73,81	73,59	16389,63
	74,56	74,63	74,52	74,39	16434,33
	74,72	74,39	74,59	74,30	16487,44

Tablo 4.2 Birinci Durum En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	73,18 / 45	74,29 / 38	73,59 / 43
Doğrulama	74,64 / 45	75,00 / 38	75,00 /43



Şekil 4.1 Birinci Durum İterasyonların Sonuçları

Elde edilen sonuçlar, α değerinin 0,5 olduğu üçüncü bölümdeki sonuçlar ile karşılaştırıldığında:

- Doğrulama verisi üzerinde ortalama nihai doğruluk, kesinlik, duyarlılık ve F1 sonuçlarında sırasıyla %1,72, %1,89, %1,68 ve %1,83 daha başarılıdır.
- Test verisi üzerinde ortalama nihai doğruluk, kesinlik, duyarlılık ve F1 sonuçlarında sırasıyla %1,03, %1,13, %1,02 ve %1,06 daha başarılıdır.
- En başarılı iterasyon anındaki ortalama doğruluk değeri, doğrulama ve test verileri için sırasıyla %1,74 ve %0,97 daha başarılıdır.

Daha başarılı sonuçlar elde edilebilmiştir ancak beraberinde 2 kat daha fazla süreye ihtiyaç duyulmuştur. Sonuçlar daha başarılı olmasına rağmen aralarında

büyük farklılık yoktur. Yani kolektiflik, risk artsa bile başarısını büyük ölçüde koruyabilmiştir. Ayrıca, aynı kısıtlar altında önerilen yöntem her iterasyonda daha az verinin sözde etiketini üretmektedir. Bu durumlar göz önüne alındığında risk almak süreden büyük ölçüde kazanç sağlarken, başarımdan hafif düşüse sebebiyet vermiştir.

4.4.2 İkinci Durum

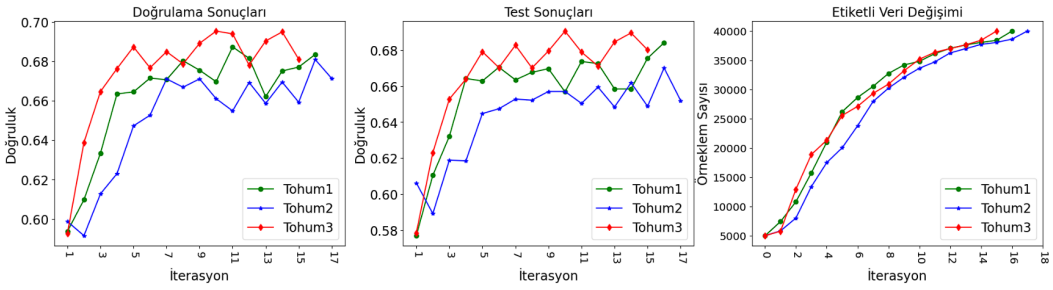
Var olan yöntem Tablo 2.3'deki öenrilen yöntemin üst satırındaki kısıtlar ile çalıştırıldığında elde edilen sonuçlar Tablo 4.3, 4.4 ve Şekil 4.2'de verilmiştir.

Tablo 4.3 İkinci Durum Nihai Sonuçları

Veri	Doğruluk	Kesinlik	Duyarlılık	F1	Süre(sn)
Test	68,41	68,44	68,76	67,85	5997,12
	65,20	65,51	65,63	64,83	6393,92
	68,01	67,16	67,94	66,96	5370,41
Doğrulama	68,35	68,50	68,46	67,76	5997,12
	67,13	67,03	67,11	66,41	6393,92
	68,10	67,33	67,93	66,92	5370,41

Tablo 4.4 İkinci Durum En Başarılı Andaki Doğruluk Sonuçları

Veri	Tohum1/Konum	Tohum2/Konum	Tohum3/Konum
Test	67,36 / 11	67,01 / 16	69,05 / 10
Doğrulama	68,73 / 11	68,09 / 16	69,53 / 10



Şekil 4.2 İkinci Durum İterasyonların Sonuçları

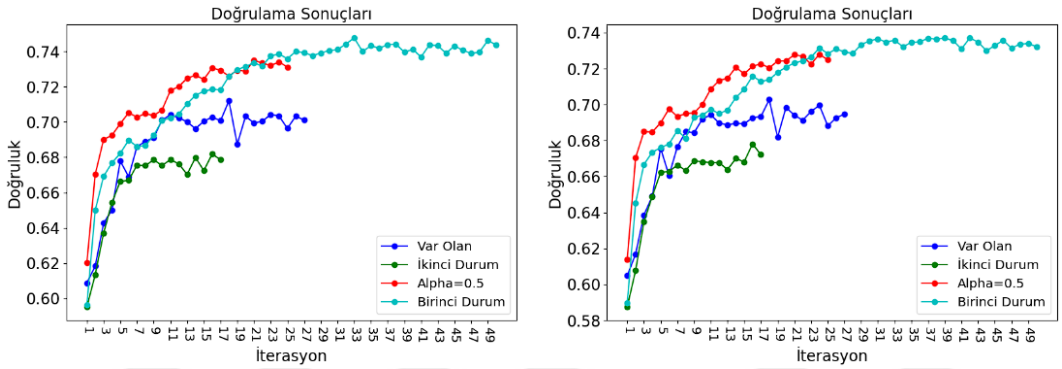
Elde edilen sonuçlar var olan yöntemin DenseNet121 mimarisindeki sonuçlar ile kıyaslandığında:

- Doğrulama verisi üzerinde ortalama nihai doğruluk, kesinlik, duyarlılık ve F1 sonuçlarında sırasıyla %3,21, %3,7, %3,2, %4,02 daha başarısızdır.

- Test verisi üzerinde ortalama nihai doğruluk, kesinlik, duyarlılık ve F1 sonuçlarında sırasıyla %3,25, %3,84, %3,27 ve %4,01 daha başarısızdır.
- En başarılı iterasyon anındaki ortalama doğruluk değeri, doğrulama ve test verileri için sırasıyla %3,73 ve %3,62 daha başarısızdır.

Her iterasyon sonrasında alınan risk fazla olduğundan daha başarısız sonuçlar elde edilmiştir. Buna rağmen ortalama nihai sonuç verme süresinde ise yarı yarıya düşüş görülmüştür.

Hem birinci durum hem de ikinci durumun her iterasyondaki ortalama değerlerinin yer aldığı doğruluk grafikleri Şekil 4.3'te verilmiştir.



Şekil 4.3 Birinci ve İkinci Durumlarındaki İterasyonların Ortalama Sonuçları

Şekil 4.2 incelendiğinde: $\alpha = 0,5$, ikinci duruma göre daha hızlı bir şekilde doygunluğa ulaşmıştır. Bu sonucun sebebi ikinci durumda her iterasyonda daha az örneklemin etiketleniyor olmasıdır.

5
DENEYSEL BULGULARIN TARTIŞILMASI

Sözde etiket üretim aşamasında geçmiş iterasyondaki modellerin tahmin olasılıklarını da dahil ederek var olan kendi kendine öğrenme algoritmasına yenilik katan önerilen yöntemin deneysel sonuçları aşağıdaki gibi özetlenmiştir:

- Sabit α sonuçları doğrultusunda en iyi α değerinin 0,8 olduğu bulunmuştur. Ancak 0,7 ve 0,9 α değerlerinde de yaklaşık aynı başarımlar elde edilmiştir. Sabit α değerlerinin sonuçları sadece CIFAR10 ve DenseNet121 ile elde edildiğinden farklı veri kümesi veya mimari altında hafif farklılıklar görülebilinir.
- α değerininin artmasına paralel olarak nihai sonucunda elde edilme süresini artmıştır. Yani eski modellerin kolektif sonucu, anlık eğitimi tamamlanan modele göre daha yüksek etkiye sahip olduğundan önerilen yöntem daha fazla süreye ihtiyaç duymaktadır.
- Adaptif α yaklaşımında ise başarım biraz daha iyileştirilebilinmiştir. Özellikle birinci iterasyon ile ikinci iterasyon arasındaki başarım farkı fazla olduğundan ikinci iterasyonda ki modelin daha baskın olması mantıklı olacaktır. Ancak, üçüncü iterasyon ve sonrasında ϕ_E ile ϕ_M arasındaki baskınlık değişecektir. ϕ_E ve ϕ_M arasındaki baskınlığa göre değişebilen adaptif α yaklaşımı daha başarılı sonuçlar vermiştir.
- Veri kümelerinde bulunan örneklemlerden bağımsızlığı sağlamak yapılan her üç tohum sonucunda da önerilen yöntem daha yüksek başarıma sahip olmuştur.
- CIFAR10 ve EfficientNetB0 sonuçları ile DenseNet121 sonuçları kıyaslandığında farklı mimarilerde de önerilen yöntemin daha başarılı sonuç verdiği gözlemlenmiştir. Ancak, veri setinin farklılaşması sadece başarım oranını etkilemiştir. Böylelikle önerilen yöntemin farklı mimariler altındaki başarımı kanıtlanmıştır.

- Intel veri kümesi ile farklı veri kümesine geçildiğinde her 2 mimaride de önerilen yöntem daha yüksek vermiştir. Ancak farklı veri kümesinden dolayı sonuçların oranlarında farklılıklar görülmüştür. Yani önerilen yöntem farklı veri kümesinde daha başarılı sonuçlar vermesine karşın var olan yönteme göre oransal farklılık göstermiştir.

- Gtzan veri kümesi ve ResNet34 mimari sonuçlarında da önerilen yöntem çok yüksek bir oranda daha başarılı çıkmıştır.

- Hem nihai sonuçlar hem de en iyi iterasyon anındaki sonuçlar dikkate alındığında: önerilen yöntem var olan yönteme göre her zaman daha başarılı sonuçlar vermiştir. Veri kümesi ve mimarinin farklılaşması sadece başarım oranını etkilemiştir.

- Var olan yöntemin her bir iterasyon sonucundaki grafikleri incelendiğinde iterasyonlar arası kırılmaların olabileceği görülmüştür. Her iterasyondaki sonuç tamamiyle anlık eğitilen modele bağlı olduğundan kırılmalar meydana gelmiştir. Önerilen yöntem de ise ani kırılmalar gözlenmemiş olup, çok daha yumuşak geçişler olmuştur.

- Var olan yöntem her iterasyonda daha fazla veriyi etiketli hale getirebişken, önerilen yöntem daha az örneklemi etiketleyebilmiştir. Bu durum göz önüne alındığında modellerin yüksek olasılıkla tahmin ettiği örneklemler arasında farklılık olduğu anlaşılmıştır. Çünkü önerilen yöntemde modellerin kolektif sonucu hesaplandığından, yüksek olasılıklı örneklemlerin farklı olması durumunda kolektif olasılık tanımlanan olasılık şartını aşamamıştır.

- Ortalama başarım grafiklerinin bazı yerlerinde var olan yöntem önerilen yönteme fazlasıyla yakın başarımlar sergilemiştir. Ancak bu durumun oluşmasında etiketlenen toplam örneklem sayısı etkili olmuştur. Önerilen yöntem her iterasyon sonunda daha az örneklem etiketleyebildiğinden ilerleyen iterasyonlarda başarımını arttırabilmişken, var olan yöntem arttıramamıştır.

6
SONUÇ

Bu çalışmada, kendi kendine öğrenme algoritmasına kıyasla yaklaşık olarak aynı süre içinde daha iyi sonuçlar elde etmek amacıyla yeni bir kolektif yaklaşım önerilmektedir. Elde edilen deneysel bulgular, önerilen yöntemin umut vadettiğini göstermektedir. Önerilen kolektif yaklaşım, mevcut yöntemle aynı kısıtlar altında çalıştırıldığında daha fazla süreye ihtiyaç duyduğundan, her iterasyon sonunda daha fazla risk alınarak ilerleme sağlanmıştır. Ayrıca, iteratif sürecin sonlandırılması için gerekli olan sözde etiket sayısı daha düşük tutulmuş ve böylece mevcut yönteme kıyasla daha kısa sürede sonuç elde edilmiştir. Önerilen kolektif yaklaşım kapsamında, sözde etiket üretim aşamasında yalnızca anlık modelin tahminleri değil, aynı zamanda önceki iterasyonlarda eğitilmiş modellerin oluşturduğu kolektif tahminler de kullanılmıştır. Bu yaklaşım, üç farklı veri seti ve derin ağ mimarisi üzerinde uygulanarak hem veri setinden hem de kullanılan mimariden bağımsız olduğu kanıtlanmıştır. Deneysel çalışmalar boyunca, anlık modelin tahmini ile geçmiş modellerden elde edilen kolektif tahmin sırasıyla $1 - \alpha$ ve α değerleriyle ağırlıklandırılarak birleştirilmiştir. Bu birleştirme sonucunda elde edilen çıktı, sözde etiket üretiminde kullanılmıştır. Farklı α değerleri ile gerçekleştirilen deneysel çalışmalarda, en iyi sonuç $\alpha = 0,8$ değeriyle elde edilmiştir. Ancak, sabit bir α değeri kullanmak yerine, doğrulama verisi üzerinde hem anlık modelin tahminlerinin hem de geçmiş iterasyonlardan üretilen kolektif tahminlerin doğruluk değerleri doğrultusunda adaptif bir α mekanizması kurulmuştur. Gerçekleştirilen deneysel çalışmalar sonucunda, önerilen kolektif yöntem, kullanılan mimari ve veri setine bağlı olarak test verisi üzerinde ortalama %5,35-%14,82 daha yüksek doğruluğa sahip olmuştur.

KAYNAKÇA

[1] C. M. Bishop, N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[2] I. Goodfellow, *Deep learning*. MIT press, 2016, vol. 196.

[3] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[4] K. Weiss, T. M. Khoshgoftaar, D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, pp. 1–40, 2016.

[5] L. Torrey, J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 2010, pp. 242–264.

[6] X. J. Zhu, "Semi-supervised learning literature survey," 2005.

[7] O. Chapelle, B. Scholkopf, A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[8] J. E. Van Engelen, H. H. Hoos, "A survey on semi-supervised learning," *Machine learning*, vol. 109, no. 2, pp. 373–440, 2020.

[9] I. Sutskever, J. Martens, G. Dahl, G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, PMLR, 2013, pp. 1139–1147.

[10] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[11] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[12] X. Dong, Z. Yu, W. Cao, Y. Shi, Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[13] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] D. Ciregan, U. Meier, J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*, IEEE, 2012, pp. 3642–3649.

[15] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, P. E. Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, vol. 14, no. 9, pp. 1360–1371, 2005.

[16] C. Garcia, M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 26, no. 11, pp. 1408–1423, 2004.

[17] M. Osadchy, M. Miller, Y. Cun, "Synergistic face detection and pose estimation with energy-based models," *Advances in neural information processing systems*, vol. 17, 2004.

[18] M. AKDİŞ, A. ELBİR, B. DİRİ, "An ensemble approach for semi-supervised self learning algorithm," in *2024 Innovations in Intelligent Systems and Applications Conference (ASYU)*, 2024, pp. 1–6. doi: 10.1109/ASYU62119.2024.10757153.

[19] M. Bianchini, M. Maggini, L. C. Jain, *et al.*, *Handbook on neural information processing*. Springer, 2013, vol. 7.

[20] A. Blum, T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[21] Z.-H. Zhou, M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 11, pp. 1529–1541, 2005.

[22] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd annual meeting of the association for computational linguistics*, 1995, pp. 189–196.

[23] E. Riloff, J. Wiebe, T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, 2003, pp. 25–32.

[24] C. Rosenberg, M. Hebert, H. Schneiderman, "Semi-supervised self-training of object detection models," 2005.

[25] I. Dópido, J. Li, P. R. Marpu, A. Plaza, J. M. B. Dias, J. A. Benediktsson, "Semisupervised self-learning for hyperspectral image classification," *IEEE transactions on geoscience and remote sensing*, vol. 51, no. 7, pp. 4032–4044, 2013.

[26] D.-H. Lee *et al.*, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, Atlanta, vol. 3, 2013, p. 896.

[27] I. Triguero, S. García, F. Herrera, "Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study," *Knowledge and Information systems*, vol. 42, pp. 245–284, 2015.

[28] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, D. Mahajan, "Billion-scale semi-supervised learning for image classification," *arXiv preprint arXiv:1905.00546*, 2019.

[29] Q. Xie, M.-T. Luong, E. Hovy, Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 687–10 698.

[30] E. D. Cubuk, B. Zoph, J. Shlens, Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[32] G. Huang, Y. Sun, Z. Liu, D. Sedra, K. Q. Weinberger, "Deep networks with stochastic depth," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, Springer, 2016, pp. 646–661.

[33] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[34] J. Böttinger. "Intel image classification." (2018), [Online]. Available: `https://www.kaggle.com/datasets/puneet6060/intel-image-classification`.

[35] G. Tzanetakis, P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[36] M. Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[37] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[38] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[39] I. Loshchilov, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

TEZDEN ÜRETİLMİŞ YAYINLAR

Konferans Bildirisi

1. M. AKDİŞ, A. ELBİR and B. DİRİ, "An Ensemble Approach for Semi-Supervised Self Learning Algorithm," 2024 Innovations in Intelligent Systems and Applications Conference (ASYU), Ankara, Turkiye, 2024, pp. 1-6, doi: 10.1109/ASYU62119.2024.10757153.