FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

AZ ÖRNEKLE ÖĞRENME PROBLEMLERİNDE DERİN ÖĞRENME TEMELLİ META-ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Muhammet ALKAN

İSTANBUL, 2020

FATİH SULTAN MEHMET VAKIF ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

AZ ÖRNEKLE ÖĞRENME PROBLEMLERİNDE DERİN ÖĞRENME TEMELLİ META-ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Muhammet ALKAN
(190221011)

Danışman
(Dr. Öğr. Üyesi Ayla GÜLCÜ)

İSTANBUL, 2020

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Bilgisayar Mühendisliği Anabilim Dalı'nda **190221011** numaralı **MUHAMMET ALKAN'**ın hazırladığı **"Derin Öğrenme Ağlarında Meta-Öğrenme Algoritmalarının Performanslarının Karşılaştırılması"** konulu Yüksek Lisans Tezi ile ilgili **TEZ SAVUNMA SINAVI,** 21/07/2020, Salı günü saat 11.00'da Çevrimiçi Video Görüşmesi ile yapılmış, sorulan sorulara alınan cevaplar sonunda adayın tezinin [**KABULÜ'**]**ne** [**OYBİRLİĞİ**] ile karar verilmiştir.

Düzeltme verilmesi halinde:
Adı geçen öğrencinin Tez Savunma Sınavı […]/[…]/20[…], tarihinde, saat da yapılacaktır.

Tez adı değişikliği yapılması halinde :

Tez adının **Az Örnekle Öğrenme Problemlerinde Derin Öğrenme Temelli Meta-Öğrenme Algoritmalarının Karşılaştırılması** şeklinde değiştirilmesi uygundur.

JÜRİ ÜYESİ	KANAATİ (*)	İMZA
Prof. Dr. A. Şima UYAR	KABUL	
Dr. Öğretim Üyesi Ayla GÜLCÜ (Danışman)	KABUL	
Dr. Öğretim Üyesi Ömer KORÇAK	KABUL	

BEYAN/ ETİK BİLDİRİM

Bu tezin yazılmasında bilimsel ahlak kurallarına uyulduğunu, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, tezin herhangi bir kısmının bağlı olduğum üniversite veya bir başka üniversitedeki başka bir çalışma olarak sunulmadığını beyan ederim.

Muhammet ALKAN
İmza

AZ ÖRNEKLE ÖĞRENME PROBLEMLERİNDE DERİN ÖĞRENME TEMELLİ META-ÖĞRENME ALGORİTMALARININ KARŞILAŞTIRILMASI

Muhammet ALKAN

ÖZET

Meta-öğrenme, literatürde daha çok öğrenmeyi öğrenme olarak dikkat çekmektedir ve bunun temel sebebi ise makine öğrenmesi yaklaşımlarının eğitim sürecini daha önceki eğitimlerden elde edilmiş olan genel özellikleri kullanarak kısaltmayı amaçlamasıdır. İnsanlardan örnek vermek gerekirse, yeni bir konuyu öğrenirken daha önceki benzer konularla ilişki kurarak önceden elde etmiş oldukları bilgiyle birlikte yeni konuyu öğrenme sürecini az sayıda örneğe bakarak başarılı bir şekilde tamamlarlar. Aynı şekilde, makine öğrenmesi algoritmalarının her defasında büyük bir veri kümesine ihtiyaç duymaksızın, az sayıda örnekle ve önceki algoritmalardan öğrenilmiş olan meta-bilgilerle yeni görevler için daha hızlı bir şekilde genelleştirilebilmesi meta-öğrenme sayesinde mümkündür.

Meta-öğrenme algoritmaları iki ana işlem içermektedir ve bu işlemler için iç içe iki döngüye sahiptir. Dışarıdaki döngüde görevler hakkında genel özellikler öğrenilmeye ve genel bilgiler çıkarılmaya çalışılırken, içerideki döngüde ise yeni gelecek olan görevlere daha çabuk adapte olmaya çalışılır. Dışarıdaki döngüde çıkartılan genel özellikler sayesinde, içerideki adaptasyon sürecinin daha kısa ve daha doğru sonuçlar elde etmesi sağlanır.

MAML ve ProtoNet gibi, literatürde karşılaştırma için çokça kullanılmakta olan meta-öğrenme algoritmalarının, az örnekle öğrenme problemlerine uygulanarak Omniglot, MiniImageNet, CIFAR100 ve CUB gibi birden fazla veri kümesi üzerinde elde edecekleri sonuçlar ayrıntılı incelendi. Bu sonuçlara bakarak meta-öğrenme

hakkında, kullanılan algoritmalar (MAML ve ProtoNet) ve veri kümeleri (Omniglot, MiniImageNet, CIFAR100 ve CUB) hakkında çıkarımlar yapıldı.

MAML algoritması için; eğitim ve test sürecindeki adım sayıları, adım genişliği gibi parametreler farklı yol sayısı (way) ve örnek sayısı (shot) yapılandırmaları üzerinde test edilmiştir. Örnek sayısı 1 olarak alındığında MAML algoritması daha başarılı sonuçlar elde ederken örnek sayısı 5 olarak alındığında ise MAML ve ProtoNet algoritmaları yaklaşık olarak benzer sonuçlar elde etmişlerdir.

Anahtar kelimeler; **meta-öğrenme, az örnekle sınıflandırma, MAML, ProtoNet.**

COMPARISON OF DEEP LEARNING BASED META-LEARNING ALGORITHMS ON FEW-SHOT LEARNING PROBLEMS

Muhammet ALKAN

ABSTRACT

Meta-learning stands out as “learning to learn” in the literature, and aims to shorten the training process of machine learning approaches by using the general features obtained from previous training. For example, while people learn a new subject, they successfully complete the process of learning the new subject with a few examples by and establishing a relationship with the previous similar topics and the knowledge they have previously obtained. Likewise, machine learning algorithms can be quickly generalized for new tasks with a few training examples and knowledge learned from previous training examples without the need for a large data set each time.

Meta-learning algorithms involve two main processes and have two nested loops for these processes. While trying to learn general features in the outer loop about the tasks and to get general information, it is tried to adapt to the new tasks more quickly in the inner loop. By this way, learned general features in the outer loop makes the adaptation process inside shorter and ensures it gets more accurate results.

Meta-learning algorithms, such as MAML and ProtoNet, which are widely used in the literature are applied to few-shot learning problems and the obtained results examined in detail on multiple data sets such as Omniglot, MiniImageNet, CIFAR100 and CUB. Based on these results, inferences about meta-learning,

algorithms (MAML, ProtoNet) and datasets (Omniglot, MiniImageNet, CIFAR100 and CUB) were made.

Parameters such as number of gradient steps and step size in the training and testing were tested on different way and shot configurations for the MAML algorithm. While MAML obtained more successful results when the number of shot is taken as 1, MAML and ProtoNet algorithms obtained approximately similar results when the number of shot was taken as 5.

Keywords; **meta-learning, few-shot classification, MAML, ProtoNet.**

ÖNSÖZ

Bu tez çalışmasında, meta-öğrenme algoritmalarının az örnekle öğrenme problemlerine uygulanarak, birden fazla veri kümesi üzerinde elde edilecek olan sonuçların karşılaştırılması yapılmıştır. Tez çalışmam süresince, benim için yeni bir konu ve heyecan olan bu süreçte desteklerini esirgemeyen, tecrübesiyle beni her seferinde yönlendiren değerli danışman hocam Dr. Öğr. Üyesi Ayla GÜLCÜ'ye teşekkürü borç bilirim.

Ayrıca yaptığımız toplantılarda ve sonrasında fikir alışverişinde bulunduğum Arş. Gör. Zeki KUŞ'a katkılarından dolayı teşekkürü borç bilirim.

Son olarak ise, yüksek lisans sürecimde sürekli olarak bana moral veren ve destekleyen bölüm başkanımız Prof. Dr. Ali Yılmaz ÇAMURCU hocama bana katmış oldukları için teşekkürü borç bilirim. Doktora ve sonrasında çalışmak istediğim bir ortam ve konu sağlanmasında yardımcı olan bütün bölüm hocalarıma teşekkür ederim.

Muhammet ALKAN

İÇİNDEKİLER

ÇİZELGE LİSTESİ

ŞEKİL LİSTESİ

KISALTMALAR

CNN	Convolutional Neural Networks
SGD	Stochastic Gradient Descent
MAML	Model-Agnostic Meta-Learning
ProtoNet	Prototypical Networks
FC100	Fewshot-CIFAR100
CUB	Caltech-UCSD Birds
MSL	Multi-Step Loss Optimization

GİRİŞ

Makine öğrenmesi yaklaşımları birçok alanda kullanılmakta olup büyük veri kümelerine (eğitim verisi) bağlı olarak sonuçlar çıkartmaktadır. Eğitim verisinin büyüklüğü modelin eğitim sürecinin uzamasına sebep olsa da eğitilecek olan modelin doğru sonuç verme oranını yükselttiği düşünüldüğü için genelde büyük veri kümeleri kullanılarak modeller eğitilir. Yeni bir görev için eğitilecek olan model her seferinde bu eğitim sürecine sıfırdan başlamak durumundadır. Görevler arasında benzerlikler bulunsa bile herhangi bir bilgi aktarımı olmadığı durumda, eğitim sürelerinde bir kısalma söz konusu olmamaktadır. Makine öğrenmesi yaklaşımlarının farklı görevler üzerinde nasıl sonuç verdiğine bakarak, bu sonuçlardan elde edilen tecrübelerin yeni görevlerin daha hızlı öğrenilmesine katkısı olup olmayacağını kararlaştırmaya meta-öğrenme veya bir başka deyişle öğrenmeyi öğrenme denir.

Meta-öğrenme, literatürde daha çok öğrenmeyi öğrenme (learning to learn) olarak dikkat çekmektedir. Öğrenmeyi öğrenme olarak dikkat çekmesinin sebebi, makine öğrenmesi yaklaşımlarının eğitim sürecini daha önceki eğitimlerden elde edilmiş olan meta-bilgilere (meta-data) bakarak kısaltmayı amaçlamasıdır. Bu yaklaşım, yeni eğitim sürecinin daha hızlı olması için gerekli altyapıyı oluşturur ve bu sayede eğitim süreci sıfırdan başlatılmamış olur. İnsanlardan örnek vermek gerekirse, yeni bir konuyu öğrenirken daha önceki benzer konularla ilişki kurarak önceden elde etmiş oldukları bilgiyle birlikte yeni konuyu öğrenme sürecini az sayıda örneğe bakarak başarılı bir şekilde tamamlarlar. Aynı şekilde, makine öğrenmesi algoritmalarının her defasında büyük bir veri kümesine ihtiyaç duymaksızın, az sayıda örnekle ve önceki algoritmalardan öğrenilmiş olan meta-bilgilerle yeni görevler için daha hızlı bir şekilde genelleştirilebilmesi meta-öğrenme sayesinde mümkündür.

Meta-öğrenmenin önemi, çok sayıda örnek gerektirmeden, doğru özellikleri daha önceki eğitimlerden elde edilmiş olan meta-bilgilere bakarak daha hızlı elde

etmesinden gelmektedir. Daha önceden öğrenilmiş olan bilgilerden yola çıktığı için, yararsız olduğu bilinen eylemleri denemekten kaçınarak eğitim sürecini daha hızlı bir şekilde sonuçlandırır. Bu sayede, yeni görevlere daha hızlı bir şekilde adapte olarak az sayıda örnekle bile yeterince iyi öğrenebilir. Bu süreç, sadece makine öğrenme modelini hızlandırmak ve iyileştirmekle kalmaz, aynı zamanda kullanılan algoritmaların veri odaklı bir şekilde öğrenilen yeni yaklaşımlarla değiştirilmesine olanak tanır.

Aynı süreç insanlar için de geçerlidir; yeni bir konuyu/işi öğrenmeye nadiren sıfırdan başlarız, genelde daha önceki birikimlerimizden faydalanarak bu öğrenme sürecini hızlandırırız. Öncelikle daha önceden öğrenmiş olduğumuz becerilere bakar, daha önce işe yaramış olduğunu gördüğümüz yaklaşımları yeniden kullanır ve bu tecrübelere dayanarak denemeye değer olan şeylere odaklanıp yararsız olduğunu bildiğimiz eylemleri denemekten kaçınırız. Bu şekilde, her yeni konu/iş öğrenildiğinde elde edilen tecrübeler sayesinde yeni konuları/işleri öğrenmek daha kolay hale gelir ve yeni öğrenme süreci daha az örnek ve daha az deneme-yanılma gerektirir. Bu sürecin temeline baktığımızda, öğrenilen konu veya işten bağımsız olarak asıl kazanımın yeni bir konuyu veya yeni bir işi "nasıl öğreneceğimizi öğrenmemiz" olduğunu görürüz. Benzer şekilde, bu öğrenmeyi öğrenme deneyiminin makine öğrenmesi için uygulanması meta-öğrenme fikrinin temelini oluşturur. Bu fikrin makine öğrenmesine uyarlanması sonucunda, belirli bir görev için makine öğrenimi modelleri oluştururken, genellikle benzer/yakın olabilecek diğer görevlerle ilgili deneyimlerin de geliştiği ve bu durumun en azından daha doğru seçimlerin yapılmasına yardımcı olacağı düşünülür. Ayrıca meta-öğrenme yaklaşımları bölüm 2'de daha detaylı olarak incelenecektir.

Otomatik nesne sınıflandırma/tanıma için literatürde çok sayıda yaklaşım üretilmiştir. Ancak bu sınıflandırmayı yapabilecek modelin eğitilmesi için gerekli olan eğitim örneklerini elde edebilmek çoğu zaman mümkün değildir veya sayıca yeteli değildir. Bunun için yapay örneklerin üretilmesi fikri ortaya atılmıştır ancak bu yöntemle gerçekçi örnekler üretilemediği için bu fikir başarılı bulunmamıştır. Birkaç sınıf nesneyi değil ama binlerce farklı sınıftaki nesnelerin hepsini birden sınıflandırma/tanıma yapmak istediğimizde karşımıza çıkan en büyük engel budur.

Her bir yeni sınıfı tanımak için binlerce örnek yerine birkaç örnek ile eğitebilecek bir sistem geliştirilmesi araştırmacılar için en büyük hedef olmuştur ve bu problemler genel olarak az örnekle öğrenme olarak bilinmektedir.

Bu tezde, MAML ve ProtoNet gibi meta-öğrenme yaklaşımlarının az örnekle öğrenme problemlerindeki başarısını ölçmeyi ve bu başarının nelere bağlı olduğunu gösterebilmeyi hedefliyoruz. MAML ve ProtoNet algoritmalarını, Omniglot, MiniImageNet, FC100 (Fewshot-CIFAR100) ve CUB (Caltech-UCSD Birds) gibi farklı veri setleri üzerinde test ederek algoritmalara bağlı olan farklı parametre ayarlarında nasıl sonuç aldıkları karşılaştırmalı olarak verilecektir. Bu sayede ileride yapacağımız çalışmalar için bir temel oluşturup sonrasında ise bu konularda literatüre katkılar sunmayı hedefliyoruz.

1. META-ÖĞRENME

Meta-öğrenme, literatürde daha çok öğrenmeyi öğrenme (learning to learn) olarak dikkat çekmektedir. Öğrenmeyi öğrenme olarak dikkat çekmesinin sebebi, makine öğrenmesi yaklaşımlarının eğitim sürecini daha önceki eğitimlerden elde edilmiş olan meta-bilgilere (meta-data) bakarak kısaltmayı amaçlamasıdır. Bu yaklaşım, yeni eğitim sürecinin daha hızlı olması için gerekli altyapıyı oluşturur ve bu sayede eğitim süreci sıfırdan başlatılmamış olur. İnsanlardan örnek vermek gerekirse, yeni bir konuyu öğrenirken daha önceki benzer konularla ilişki kurarak önceden elde etmiş oldukları bilgiyle birlikte yeni konuyu öğrenme sürecini az sayıda örneğe bakarak başarılı bir şekilde tamamlarlar. Aynı şekilde, makine öğrenmesi algoritmalarının her defasında büyük bir veri kümesine ihtiyaç duymaksızın, az sayıda örnekle ve önceki algoritmalardan öğrenilmiş olan meta-bilgilerle yeni görevler için daha hızlı bir şekilde genelleştirilebilmesi meta-öğrenme sayesinde mümkündür.

Meta-öğrenme terimi, önceki görevlerden elde edilen deneyimlere dayanarak seçimler yapan her türlü öğrenmeyi kapsar. Önceki görevler ne kadar benzer olursa, elde edilecek kullanışlı meta-bilgi türleri de o kadar fazla olur. Bu görev benzerliği, elde edilecek sonucu iyileştireceği için doğru bir şekilde tanımlanması gerekmektedir. Eğer bu benzerlik tanımı doğru bir şekilde yapılmazsa, yeni görevin önceki görevlerden tamamen ilgisiz olduğu durumlarda, önceki deneyimlerden yararlanılması bir fayda sağlamayacağı gibi başlangıç durumunu da kötüleştirebilir.

1.1. LİTERATÜRDEKİ META-ÖĞRENME TEKNİKLERİ

Literatürde bulunan Meta-Öğrenme teknikleri, genelde, elde edilen meta-bilgilerin türüne göre kategorize edilir (Vanschoren, 2018).

- Model Değerlendirmelerinden Öğrenme
- Görev Özelliklerinden Öğrenme
- Önceki Modellerden Öğrenme

1.1.1. Model Değerlendirmelerinden Öğrenme

Bu teknik daha çok, yeni bir görev için uygun olabilecek konfigürasyonu öngören bir meta-öğrenici (meta-learner) eğitilmek istenildiği zaman tercih edilir. Bahsedilen uygun konfigürasyon içerisinde, hiper-parametre ayarları ve kurulan ağ mimarisinin bileşenleri gibi özellikler bulunmaktadır. Meta-öğrenici, bu özellikler üzerinde eğitilerek yeni görevler için de önerilen konfigürasyonu tahmin edebilecek duruma getirilir.

1.1.1.1. Görevden Bağımsız Öneriler

Bu tür yaklaşımlar, yeni bir görev için herhangi bir bilgi sahibi olunmadığı durumlarda uygulanır. Daha önceden karşılaşılmamış yeni bir göreve uygun olabilecek en iyi konfigürasyonu bulmak için, basit bir şekilde, daha önceden en iyi olduğu görülmüş k tane konfigürasyon seçilerek, her bir konfigürasyon sırasıyla yeni görev üzerinde değerlendirilir (Brazdil vd., 2003). Bu değerlendirme süreci, belirlenmiş olan kısıt veya kısıtlar sağlandığında durdurulur. Değerlendirme sürecini durduracak kısıtlara örnek olarak şunları verebiliriz; k sayısı için belirlenen bir değere ulaşmak, belirtilen bir zaman süresine ulaşmak veya istenilen değerlere uygun doğru bir model bulmak. Değerlendirme süreci nihayete erdiğinde, verilmiş olan yeni göreve uygun olabilecek konfigürasyonlar çıkan sonuca göre sıralanır.

1.1.1.2. Konfigürasyon Uzayı

Daha iyi bir model konfigürasyonu öğrenebilmek için önceki değerlendirmeler kullanılabilir. Daha önceden karşılaşılmamış yeni bir görev için, yeni görevden bağımsız olmasına rağmen, önceki modellerden elde edilen konfigürasyon alanının sadece daha ilgili bölgelerine bakılması, optimal model arayışını hızlandırabilir. Hesaplamalı kaynakların sınırlı olduğu durumlarda, konfigürasyon alanının kısıtlanarak sadece daha ilgili bölgelerine bakılması daha az yük getireceği için çok önemlidir.

1.1.1.3. Konfigürasyon Transferi

Belirli bir görev için önerilerde bulunmak istiyorsak, önceki görevlere ne kadar benzer olduğu konusunda ek bilgiye ihtiyaç duyarız. Bu benzerlik için, önerilen veya rastgele seçilen konfigürasyonları yeni görev üzerinde değerlendirip,

elde edilen sonuçların benzer olup olmadığına bakabiliriz. Bu sayede benzer olan görevleri öğrenebiliriz.

Benzerlik için dikkate alınan özelliklere örnek olarak şunları verebiliriz; aynı görev için elde edilen performans sonucu veya aynı görev için elde edilen hata oranı gibi. Bu özelliklere bakarak, teorik olarak, farklı görevlerde aynı performansı veya aynı hata oranını elde ediyorsak, bu farklı görevlerin birbirine benzer olduğu sonucuna varabiliriz.

1.1.1.4. Öğrenme Eğrileri

Sadece eğitim sürecine bakarak, eğitim sürecinin gidişatı hakkında da meta veriler çıkartabiliriz. Eğitim sürecinin öğrenme eğrisi elde edilerek sonraki süreç hakkında çıkarımlar yapılabilir. Bunun için, modelin elde ettiği sonuçların daha fazla eğitim verisi eklendikçe ne kadar hızlı geliştiğine bakılabilir. Veya eğitim sürecini belli adımlara bölerek verilen konfigürasyon için her bir adımın elde ettiği sonuçlara bakılabilir. Bu gibi yöntemlerle öğrenme eğrisi elde edilebilir. Meta-öğrenmede, öğrenme eğrisi bilgileri görevler arasında aktarılır (Leite & Brazdil, 2010).

Yeni bir görev üzerinde denenecek olan bir konfigürasyonun ne kadar iyi sonuç verebileceğini değerlendirirken, eğitimin bütünüyle tamamlanmasını beklemeden, belirlenen adım sayısından sonra eğitimi durdurabilir ve bu konfigürasyonun bütün veri kümesine uygulandığında ne kadar iyi performans göstereceğini tahmin etmek için kısmen elde edilmiş olan öğrenme eğrisini kullanabiliriz. Diğer görevlerle ilgili önceki deneyimlere dayanarak, kısmen elde edilmiş olan bu öğrenme eğrisine bakarak eğitime devam edip etmeme konusunda karar verebiliriz. Bu süreç, eldeki veri kümesi için en uygun olabilecek konfigürasyonu arama sürecini önemli ölçüde hızlandırabilir.

1.1.2. Görev Özelliklerinden Öğrenme

Bir başka meta-veri kaynağı, eldeki göreve ait karakteristik özelliklerdir (meta-özellikler). Her bir görev, k tane meta-özellik barındıran bir vektör ile gösterilirse, bu vektör gösterimi kullanılarak görevlerin birbirine ne kadar benzediği hesaplanabilir. Benzerlik hesaplaması olarak, iki vektör arasında basit bir Öklid uzaklığı tanımlamak bile yeterli olacaktır. Öklid mesafesine dayalı olarak tanımlanan

bu görev benzerliği ölçüsü sayesinde, benzer görevlerden yeni görevlere bilgi aktarabiliriz.

1.1.2.1. Meta-Özellikler

Eldeki veri kümesi hakkında ayırt edici olabilecek olan özellikleri kullanmak performans bakımından olumlu sonuçlar getirecektir. Literatürde yaygın olarak kullanılmakta olan, genel amaçlara hitap eden meta-özellikler bulunmaktadır.

Meta-özelliklerin ayırt ediciliği veya sonuca etkisi uygulamadan uygulamaya değişiklik göstermektedir. Bütün uygulamalar için geçerli olabilecek bir meta-özellik olmadığı gibi, hangi meta-özelliğin hangi uygulamalarda daha iyi sonuç elde edebileceği denemeler sonucunda ortaya çıkacaktır. Literatürde yaygın olarak kullanılan meta-özelliklere ek olarak, bu özelliklerin kombinasyonu veya istatistiksel hesaplamalar sonucu elde edilecek başka meta-özellikler de oluşturulabilir (Kalousis & Hilario, 2000).

1.1.2.2. Meta-Özellikleri Öğrenme

Bir önceki başlık altında verildiği gibi, sabit olarak tanımlanan meta-özellikleri kullanmaktansa görevlerden elde edilecek olan bilgilerden yola çıkarak meta-özellik öğrenme yoluna gidilebilir. Örneğin, görevler aynı girdi özelliklerine sahipse (aynı çözünürlükteki görüntüler gibi), temsili bir meta-özellik öğrenebilmek için Siyam ikizi ağları da kullanılabilir (Kim vd., 2017).

1.1.2.3. Benzer Görevlerden Sıcak-Başlangıç Optimizasyonu

Görev benzerliklerini tahmin etmek ve benzer görevler için iyi sonuç verebilecek konfigürasyonlarla optimizasyon süreçlerini başlatmak (warm-starting) için meta-özellikler kullanılabilir. Meta-özelliklerin bu şekilde kullanımı, ilgili görevler hakkında deneyim verildiğinde konusunda uzman bir insanın, iyi olabileceğini düşündüğü modeller için manuel olarak arama yapmasına benzetilebilir.

1.1.3. Önceki Modellerden Öğrenme

Bir başka meta-veri kaynağı, önceki modellerin kendisidir. Önceki modelleri dikkate alıp bu modellerin yapısından ve öğrendiği parametrelerden yola çıkarak öğrenmeyi gerçekleştirebiliriz. Bu öğrenmeyi gerçekleştirmek için bir meta-öğrenici eğitilir. Bu meta-öğrenici, daha önceki benzer görevlerden ve bu görevler için kullanılmış olan modellerden yola çıkarak bir başka öğrenicinin (base-learner) yeni bir görevi başarıyla tamamlayabilmesi için nasıl eğitilmesi gerektiğine karar verir, bir başka deyişle nasıl eğitilmesi gerektiğini öğrenir.

1.1.3.1. Öğrenme Transferi

Bu yöntemde (Thrun & Pratt, 1998), bir veya birden fazla görev üzerinde eğitilmiş olan modeller temel alınarak, başka benzer görevler için model oluşturulması gerektiği durumlarda bu temel (eğitilmiş olan modeller) başlangıç noktası olarak kullanılır. Özellikle ImageNet gibi büyük veri kümelerinde önceden eğitilmiş modellerin kullanılmasının, diğer görevlere de iyi bir şekilde öğrenme transferi gerçekleştirdiği yapılan çalışmalarla ortaya koyulmuştur (Donahue vd., 2014). Fakat, bu yaklaşımın, yeni gelen görevlerin daha önceki görevlere benzer olmadığı durumlarda iyi sonuç vermediği de literatürde yapılan çalışmalarla gösterilmiştir (Yosinski vd., 2014).

1.1.3.2. Az Örnekle Öğrenme (Few-Shot Learning)

Model-Agnostik Meta-Öğrenme (MAML), benzer görevlere daha iyi genelleşen model parametrelerini başlangıçta oluşturmayı (W_{init}) öğrenir (Finn vd., 2017). Rastgele bir başlangıç ağırlıklarıyla $\{w_k\}$ başlayıp, sürekli olarak belirli sayıda görevi ele alarak, her biri görev için öğrenciyi K adet eğitim örneğini kullanarak eğitir ve test örnekleri üzerinde gradyanı ve kaybı hesaplar. Sonrasında ise, ağırlıkları $\{w_k\}$ bir sonraki görevlere daha uygun olacak bir başlangıç noktası yönünde güncellemek için meta-gradyanı geriye doğru hesaplama işlemini yapar. Başka bir deyişle, her bir görev setinden sonra, ağırlıklar $\{w_k\}$ herhangi bir görevin başlangıç durumundaki ağırlıkları $\{W_{init}\}$ için daha iyi bir hale gelir.

Reptile, MAML algoritmasının yaklaşık versiyonu olarak ortaya çıkmıştır (Nichol vd., 2018). Belirli bir görev için K adım stokastik gradyan inişini işletir ve

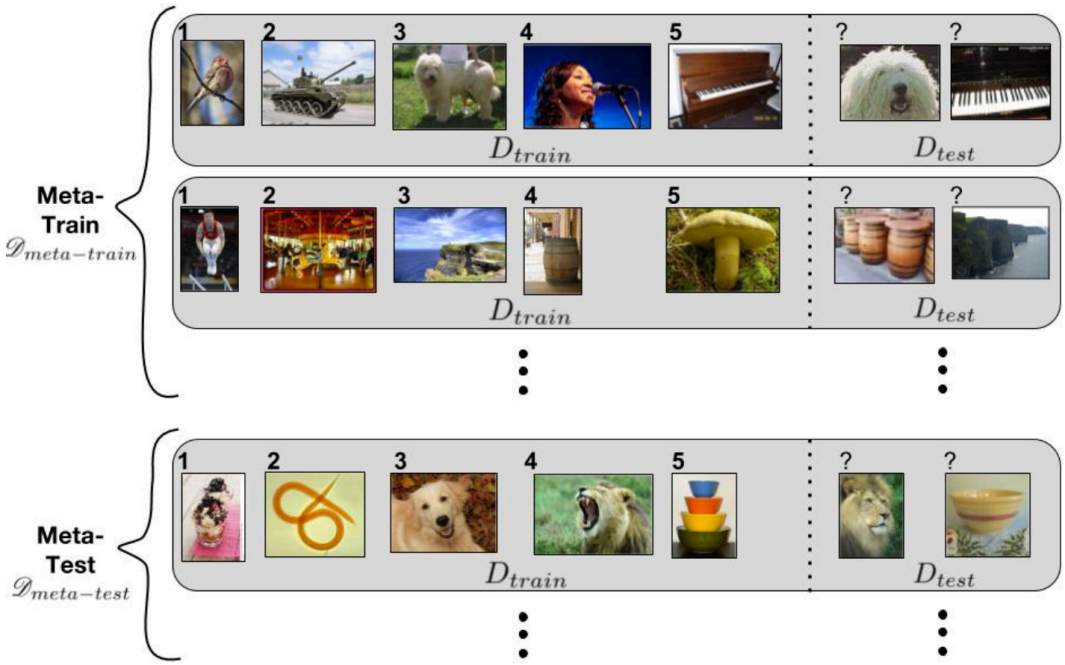
daha sonrasında bu K adımdan elde edilen ağırlıklar yönünde yavaş yavaş başlangıç ağırlıklarını hareket ettirir. Bu yaklaşık hesaplamanın temelinde şu düşünce vardır, her göreve muhtemelen birden fazla uygun olabilecek ağırlık $\{w_i^*\}$ vardır ve buradaki hedef ise her görev için bu ağırlıklardan $\{w_i^*\}$ en az birine yakın bir başlangıç ağırlıkları $\{W_{init}\}$ bulmaktır.

1.2. META-ÖĞRENMENİN İNCELENMESİ

Meta-öğrenme algoritmalarına geçmeden önce, genel olarak meta-öğrenme yaklaşımlarını incelememiz gerekmektedir. Çünkü meta-öğrenme yaklaşımlarının klasik öğrenme yöntemlerinden farklı bir yapısı vardır. Meta-öğrenme yaklaşımlarını kendi içerisinde temel iki parçaya ayırarak inceleyebiliriz; meta-öğrenme ve adaptasyon.

Meta-öğrenme yaklaşımları, klasik öğrenme yöntemlerinden (gözetimli öğrenme gibi) daha farklı bir eğitim ve test süreci barındırmaktadır. Ayrıca bu süreçler içerisinde kullanılan terminoloji de birçok yerde farklılıklar göstermektedir. Bu farklılıkların karışıklığa sebep olmaması açısından, kullanılan terimlerden de açıkça bahsedilmesi gerekmektedir.

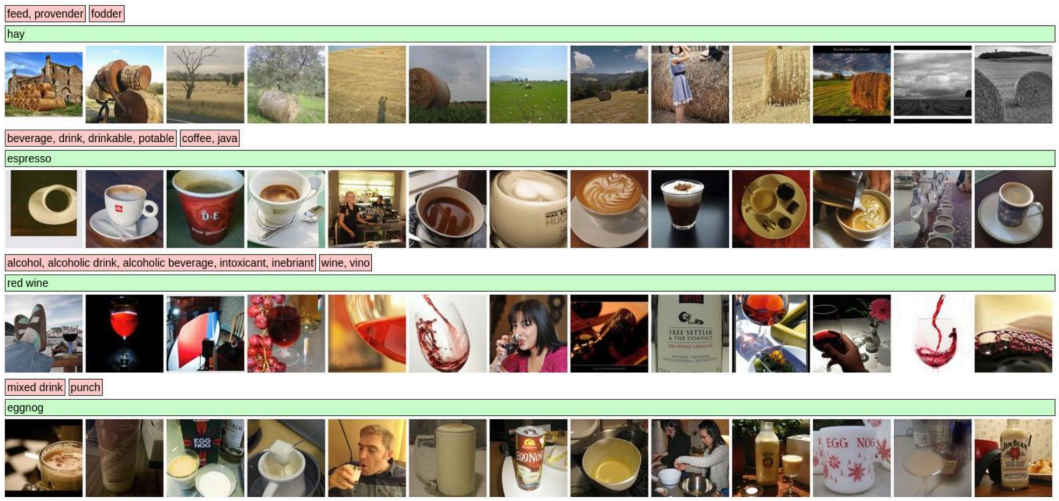
K-shot için, model herhangi bir görevi K örnekten yola çıkarak öğrenmeye, adapte olmaya çalışır. Bu öğrenme sürecinde, herhangi bir T görevi için, o görevde bulunan K adet eğitim verisi kullanılarak model eğitilir. Sonrasında ise o görevde bulunan test verileri üzerinde eğitilen bu model test edilir. Çıkan test hatası (loss) dikkate alınarak modelin parametreleri güncellenir. Yani, her bir görevin test hatası, meta-öğrenme sürecinin eğitim hatası olarak işlev görür ve bu şekilde model parametreleri güncellenir. Şekil 1.1'de yukarıda anlatılan bütün düzenin genel bir görüntüsü bulunmaktadır, eğitim ve test süreçlerini de içerecek şekilde.



Şekil 1.1: Eğitim ve test ayırımları, genel bir örnek görünüm (Kaynak: Ravi & Larochelle, 2017).

- Eğitim Veri Kümesi

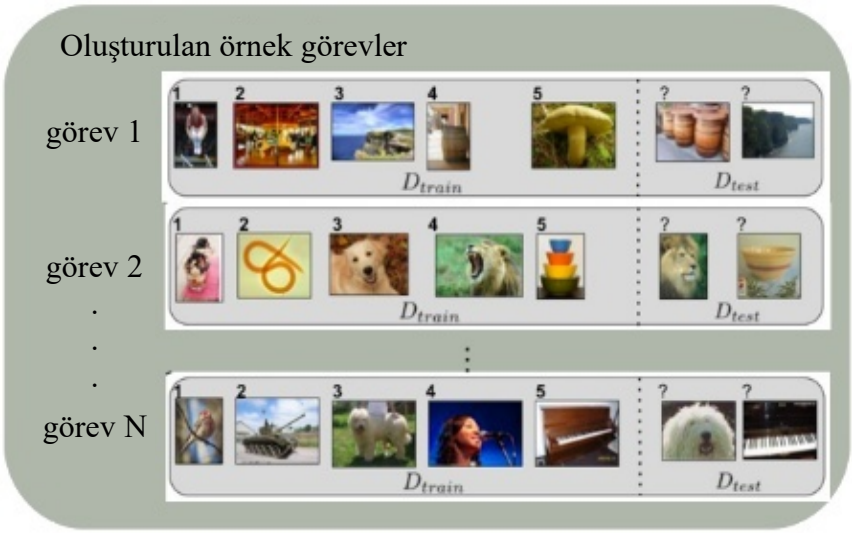
Meta-öğrenme sürecindeki eğitim görevlerinde kullanılacak olan veri kümeleridir. Algoritma bu veri kümelerindeki verilerden öğrenmeyi öğrenecektir. Aynı şekilde test ve doğrulama (validation) için de ayrı veri kümeleri bulunmaktadır.



Şekil 1.2: Örnek bir veri kümesi içerisindeki görüntüler, etiketleri ile birlikte.

- Görev (T, Task)

Her görev, destek (support) ve sorgu (query) olmak üzere iki küçük veri kümesi içermektedir. Bu küçük veri kümeleri, verilen parametreler de göz önüne alınarak, Şekil 1.2'de küçük bir örneği verilmiş olan benzer şekildeki ana eğitim veri kümesinden rastgele olarak seçilir. Klasik öğrenme yöntemlerinde bu görev tanımı bulunmamakta ve her görev için model test edilmemektedir. Örnek olarak Şekil 1.3'de verilen görevlerin tamamının içerisinde bulunan görüntülerin Şekil 1.2'de örneği verilmiş olan ana veri kümesini oluşturuyor olduğunu söyleyebiliriz.



Şekil 1.3: Veri kümesinden farklı görev setlerinin oluşturulması.

Tek bir görevin içeriği ise Şekil 1.4'teki gibi olmaktadır. N adet sınıf için her bir sınıfa ait K adet görüntüden oluşan veri grubu "support" olarak isimlendirilir ve o görevin öğrenilmesi için kullanılır. Karşılığında ise z tane (genelde 15 adet) test görüntüsü bulunmaktadır ve test için kullanılan bu veri grubu ise "query" olarak isimlendirilir. Şekil 1.4'te bir görevin örnek içeriği gösterilmeye çalışılmıştır.

Şekil 1.4: Örnek bir görev (task) içeriği.

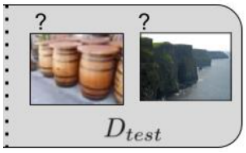
- Eğitim (D_T^{train}, support set)

Herhangi bir T görevi için seçilen eğitim verileri, genellikle eğitim veri kümesinden seçilen K adet veriden oluşur. Şekil 1.4'te verilen örnek için konuşursak, bu görev örneğinin noktalı çizgilerle ayrılmış olan sol tarafı eğitim için kullanılacaktır. Bu kısım Şekil 1.5'te ayrılarak gösterilmeye çalışılmıştır.

Şekil 1.5: Örnek görevin içerisindeki eğitim verileri.

- Test (D_T^{test}, query set)

Herhangi bir T görevi için seçilen test verileri. Yeni gelen verilerde doğru sonuçlar elde edebilmek için modelin nasıl optimize edildiği önemlidir. Bu optimizasyon sürecinden en iyi faydayı elde etmek için eğitim sürecindeki (meta-training) her bir görev için eğitim verilerinin yanında ayrıca test verileri de ayırmamız gerekiyor. Her bir görev için, öğrenilecek olan veri kümelerinin yanında ayrıca bu kategoriye karşılık gelen ve eğitim sürecinde bulunmayan test kümelerine de ihtiyacımız olacaktır. Bu test kümeleri, eğitim sürecinde kullanılan veri kümelerinden hariç tutulacaktır. Şekil 1.4'te verilen örnek için konuşursak eğer, bu görev örneğinin noktalı çizgilerle ayrılmış olan sağ tarafı test için kullanılacaktır. Bu kısım Şekil 1.6'da ayrılarak gösterilmeye çalışılmıştır.



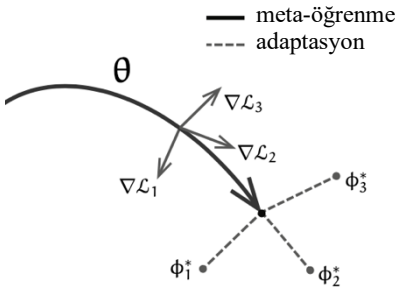
Şekil 1.6: Örnek görevin içerisindeki test verileri.

1.2.1. Adaptasyon

Test veri kümesi hakkında çıkarım yapmak istediğimiz zaman, şimdiye kadar öğrenilmiş olan veri kümelerinden (meta-training datasets) elde edilmiş olan ağırlıklarla yeni bir eğitime başlarız ve bu ağırlıkları elimizdeki test veri kümesine uygun hale getirmeye çalışırız. Bu süreç adaptasyon olarak adlandırılır. Bu sayede,

önceki eğitimlerden elde edilen tecrübeleri (ağırlıklar) kullanarak başlanılan eğitim süreci daha hızlı ve daha doğru sonuç verir. Buradaki tek kısıt; gelen test veri kümesinin daha önce öğrenilmiş olan veri kümeleriyle görev bakımından benzerlikler taşımasıdır. Bu benzerlik ne kadar iyi olursa, adaptasyon süreci de o kadar iyi sonuç verir.

Şekil 1.7'de gösterilen θ modelin parametrelerini temsil ederken siyah kalın çizgi ise meta-öğrenme sürecini temsil etmektedir. Şekil 1.7'de verildiği gibi 3 farklı görevimiz olduğunda, her görev için gradyan adımı atılır (gri çizgiler). Modelin parametrelerinin (θ), görev 1, 2 ve 3'ün tüm 3 optimal parametresine de (Φ_1, Φ_2, Φ_3) yakın olduğunu görebiliriz. İyi parametrelerle başlangıç (θ) durumu sayesinde farklı yeni görevlere hızla adapte olabilmeyi sağlar.



Şekil 1.7: Adaptasyonun temsili gösterimi (Kaynak: Finn et al., 2017).

1.2.2. Meta-Öğrenme

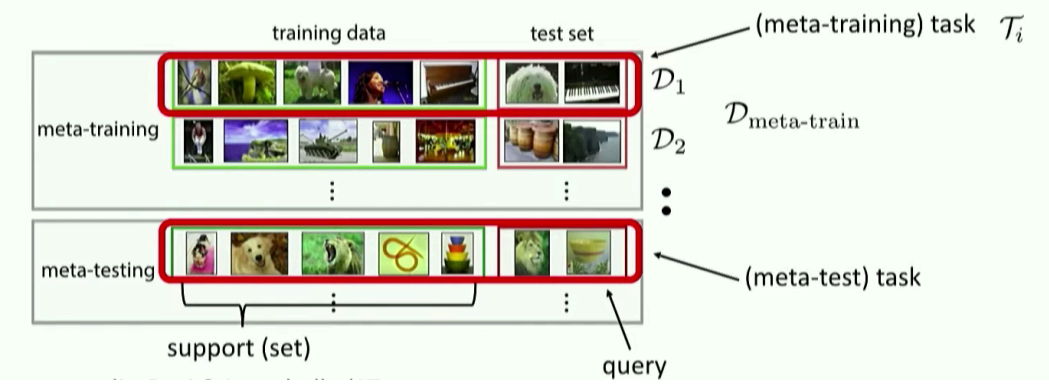
Test veri kümeleri için iyi bir başlangıç noktası sağlamak en önemli kıstas olduğundan, başlangıç parametreleri büyük önem taşımaktadır. Görevlere uygun başlangıç parametreleri sağlamak için eğitim ve test süreçleri birbiriyle örtüşmelidir. Bu örtüşme, meta-öğrenmenin en temel kuralıdır. Eğer test sürecinde gelen veri kümelerinden öğrenmek istiyorsak, öğrenilmiş olan veri kümelerinin eğitim sürecinde (meta-training) modelimizi öğrenmesi için eğitmeliyiz. Bu sayede, benzer görevler için, yeni bir görevin nasıl öğrenileceğini öğrenmiş oluruz. Adaptasyon sürecindeki amaç Fi'nin (Φ) optimize edilmesi iken meta-öğrenme sürecindeki amaç ise Teta'nın (Θ veya θ) optimize edilmesidir. Bu optimizasyon ise iki ana döngü ile elde edilmeye çalışılır. Dış döngüde (outer loop) modeli görevler üzerinde genelleştirmeye çalışıp Teta'nın (Θ veya θ) optimize edilmesi istenirken iç döngüde

ise görevler üzerindeki adaptasyon sağlanarak Fi'nin (Φ) optimize edilmesi istenir. Dışarıdaki döngüde görevler hakkında genel özellikler öğrenilmeye ve genel bilgiler çıkarılmaya çalışılırken, içerideki döngüde ise yeni gelecek olan görevlere daha çabuk adapte olmaya çalışılır. Dışarıdaki döngüde çıkartılan genel özellikler sayesinde, içerideki adaptasyon sürecinin daha doğru sonuçlar elde etmesi sağlanır. Bu işlem düzeni, aşağıda verilen dört adımın tekrarı olarak gösterilebilir.

1. Bir veya birden fazla görev örneklenir
2. Bu görevler için gerekli olan veriler örneklenir
3. Fi optimize edilir (iç döngü)
4. Teta güncellemesi yapılır (dış döngü) ve 1. adımdan devam edilir

İlk adımda gerekli olan görevler örneklenir ve yapılar oluşturulur, sonrasında ise 2. adımda bu görevler için gereken veri örnekleri rastgele olarak ana veri kümesinden seçilerek algoritma için gerekli olan yapının kurulması sağlanır. Artık modele bir döngü boyunca verilecek olan görevler elde edilmiştir. Bu görevler sırasıyla iç döngü içerisinde oluşturulan kopya modele verilerek kopya model eğitilir ve görev içerisinde bulunan test verileri üzerinde eğitilen bu kopya model test edilir. Bu sayede 3. adımdaki Fi optimizasyonu gerçeklenmiş olur. Bir döngü için örneklenmiş olan bütün görevler bu şekilde bitirildikten sonra artık dış döngüye geçilir ve iç döngüdeki kopya modelin kayıplarına göre Teta güncellemesi yapılarak dış döngüdeki meta-öğrenici (meta-learner) güncellenir. Bu sayede bir iç ve dış döngü süreci tamamlanır ve 1. adımdan itibaren aynı süreç tekrarlanır.

Meta-öğrenmede; eğitim, doğrulama ve test için farklı veri kümeleri vardır (sırasıyla $D_{meta\text{-}train}$, $D_{meta\text{-}validation}$ ve $D_{meta\text{-}test}$). $D_{meta\text{-}train}$'de, bir öğrenme prosedürü (meta-learner) eğitmekle ilgileniyoruz. Bu meta-öğrenici, girdi olarak eğitim kümelerinden birini alır ve aldığı eğitim kümesine karşılık gelen test kümesi üzerinde yüksek sınıflandırma performansı elde eden bir sınıflandırıcı (learner) üretmeye çalışır.



Şekil 1.8: MAML için örnek bir veri ayırımı ve terminolojilerin gösterilmesi

Örneğin, MAML N-way K-shot sınıflandırma görevi için, meta-eğitim veri kümesinden rastgele olarak N farklı sınıf seçilir, sonrasında ise seçilmiş olan her bir sınıf için K tane resim eğitim için ve 15 tane resim ise test için yine farklı olacak şekilde rastgele seçilir. Örnek bir veri kümesi ayırımı Şekil 1.8'de verilmiştir. Bu şekilde, toplamda N * (K + 15) tane resimden bir görev (task) oluşturulmuş olur. 4 task içeren her bir batch sonrasında meta-öğrenicinin ağırlıkları güncellenir.

1.2.3. Klasik Öğrenmeden (CNN) Farkları

Evrişimsel sinir ağları (CNN), görüntüleri modellemek için standart araçlar haline gelmiş ve sınıflandırma (Krizhevsky vd., 2017) gibi birçok görsel tanıma görevlerinde çok iyi sonuçlar elde etmiştir. ImageNet (Russakovsky vd., 2015) gibi büyük ölçekli ve etiketli veri kümelerinin bu başarıda anahtar rol oynadığı görülmektedir. Fakat bu şekilde bir veri kümesi oluşturabilmek her zaman mümkün değildir ve istenilen göreve bağlı olarak etiketlenmesi fazlaca zaman almaktadır. Özellikle sinir ağları (neural networks), iyi bir şekilde genelleme yapabilmeleri için büyük miktarda etiketlenmiş veri kullanılarak eğitilmelidir. Bu sebeplerden ötürü, derin sinir ağlarının genelleme yeteneklerini geliştirmek ve büyük etiketli veri kümelerine olan ihtiyacı ortadan kaldırmak son derece önemlidir.

Derin öğrenme yaklaşımları, yeterli miktarda etiketlenmiş veri bulunduğunda iyi performans elde etmiş olsa da (He vd., 2016), gerekli veri miktarını azaltarak sadece birkaç etiketli örnekten yeni kavramlar öğrenmek için yeterli olmadıkları çalışmalarla gösterilmiştir (Finn vd., 2017). Bir sınıflandırıcının ilk olarak orta büyüklükteki etiketlenmiş bir veri kümesi üzerinde eğitildiği ve daha sonrasında ise

yeni sınıflara uyum sağlama yeteneğinin değerlendirildiği durumlar çok az örnekle sınıflandırma (few-shot classification) olarak adlandırılır (örnek sayısı genellikle 1 veya 5 olarak alınır). Ne yazık ki, çok az örnek içeren yeni bir sınıflandırma görevi verildiğinde evrişimsel sinir ağının adaptasyonunun (fine-tuning) kötü sonuçlar elde ettiği gösterilmiştir (Finn vd., 2017; Ravi & Larochelle, 2017), bu da meta-learning gibi özel yaklaşımların ortaya çıkmasının ana sebebidir.

2. AZ ÖRNEKLE ÖĞRENME PROBLEMLERİNDE META-ÖĞRENME

Nesne tanıma işlevi (object recognition) görsel sistemimiz tarafından yapılan en iyi işlevlerdendir. Sadece bir bakışta gördüğümüz bir nesnenin ya da bir ortamın ne olduğunu ya da ne ile ilgili olduğunu söyleyebiliyoruz. Sadece "araba", "kadın" gibi sınıfsal ayırımları değil "benim arabam" ya da "benim annem" gibi kişiye özel sınıflandırmayı da otomatikman yapabiliyoruz. Sadece 6 yaşına geldiğimizde beynimiz 104 farklı nesneyi tanıyabilir durumda oluyor (Biederman, 1987). Yıllar içinde, yeni öğrendiğimiz bilgiler beynimizde eski öğrendiklerimiz ile birleşerek anlamlı bir yapı oluşturuyor. Bu yapı sayesinde yeni öğrenilen her bilgi bir sonraki yeni bilginin öğrenilmesini kolaylaştırıyor. İnsanoğlunda doğuştan var olan bu yeteneklerin benzerlerinin makinalara da öğretilmesi bu alanda çalışan bilim adamlarının en büyük hayali olmuştur. Ya da diğer açıdan bakılırsa, makinaların öğrenmelerinin insanların öğrenmesi gibi olamayışı bu alanda çalışan bilim adamları için en büyük engel olmuştur.

Otomatik nesne sınıflandırma/tanıma için literatürde çok sayıda yaklaşım üretilmiştir. Tüm bu çalışmaların sonucunda ortaya şu gerçek çıkmıştır: "Gerçek hayatta nesneler çok değişik görünümde/şekilde karşımıza çıkmaktadırlar. Bu nedenle oluşturulacak modelin yüzlerce, belki de binlerce parametresi olmalıdır. İstatistiksel olarak da bu kadar fazla parametresi olan bir model bu parametre sayısından kat be kat fazla eğitim örneğine (training example) ihtiyaç duyar" (Fei-Fei vd., 2006). Ancak bu kadar fazla eğitim örneği elde etmek çoğu zaman mümkün değildir. Bunun için yapay örneklerin üretilmesi fikri ortaya atılmıştır ancak bu yöntemle gerçekçi örnekler üretilemediği için bu fikir başarılı bulunmamıştır. Birkaç sınıf nesneyi değil ama binlerce farklı sınıftaki nesnelerin hepsini birden sınıflandırma/tanıma yapmak istediğimizde karşımıza çıkan en büyük engel budur. Her bir yeni sınıfı tanımak için binlerce örnek yerine birkaç örnek ile eğitebilecek bir sistem geliştirilmesi araştırmacılar için en büyük hedef olmuştur. İlk olarak (Fei-Fei vd., 2006) tarafından ortaya atılan fikrin temelinde de bu yatmaktadır. Ortaya attıkları hipotez ile, hali hazırda var olan etiketli binlerce eğitim datasıyla çok sınırlı sayıda kategori için eğitilen bir sistemdeki bilginin bir kısmının yeni kategorilerin

öğrenilmesi için kullanılabileceğini ve bunun da sıfırdan eğitime başlayacak bir sisteme kıyasla çok daha başarılı olacağını savunmuşlardır. 4 nesne kategorisine sahip Caltech4 (Fergus vd., 2003; Weber vd., 2000) ve 101 nesne kategorisine sahip Caltech101 veri kümelerini kullanarak yaptıkları bu alandaki ilk çalışmalarında çok az sayıda eğitim örneğiyle ilerisi için umut vadeden sonuçlar elde etmişlerdir. Ancak elde edilen sonuçlara bakıldığında önerilen yöntem pratikte kullanılabilir olmaktan çok uzaktır. Literatüre tek seferde öğrenme (one-shot learning) olarak geçen bu probleme daha sonra (Lake vd., 2011) çözüm getirmeye çalışmışlardır. Öğrenmenin önceki öğrenmelerden bağımsız gerçekleşmediğini, öğrenmenin genelleştirilebilir olması gerektiğini, bunun için hangi özelliklerin önemli olduğunun ortaya çıkarılması gerektiğini vurgulamışlardır.

Önceden edinilen soyut bilginin sonradan gerçekleşen öğrenme sürecinde kullanılması kavramı, öğrenmeyi transfer (transfer learning), temsil öğrenme (representation learning) ya da öğrenmeyi öğrenme (learning to learn) gibi isimlerle anılmaktadır. Dikkat ile öğrenmenin (attentional learning) (Smith vd., 2002) ve aşırı hipotezlerin (overhypotheses) (Kemp vd., 2007) soyut bilgilerin edinilmesinde önemli etken olduğu ortaya atılmıştır. Beyinde belli bir boyuta göre net bir şekilde organize edilen bilgilerin daha sonra yeni bilgiler edinirken öğreniciyi o boyuta yoğunlaştırdığı ortaya konmuştur. Ancak bu yaklaşım, öğrenmeye etki edecek boyutların önceden ortaya konmasını gerektirmesi açısından pratikte kullanılabilirliği zordur. Üzerinde en çok çalışılan nesne sınıflandırma veri kümelerinden standart MNIST veri kümesi için (LeCun vd., 1998) %99 üzerinde başarı sağlanmış olsa da bu başarı tek-seferde öğrenme probleminin çözümüne aktarılabilir değildir çünkü MNIST veri kümesi her bir kategori için binlerce eğitim örneği içermektedir. Hâlbuki ki insanlar bazen tek bir eğitim örneği ile yeni bir karakteri sınıflandırmayı öğrenebilirler.

2.1. VERİ KÜMELERİ

Meta-öğrenme algoritmalarının değerlendirilmesinde, literatürde sıkça karşımıza çıkmakta olan iki adet veri kümesi vardır. Birincisi, 50 alfabeden yaklaşık 1600 karakterin oluşturduğu ve her bir karakter için 20 adet görüntü içeren Omniglot veri kümesidir. Bu görüntüler 28x28 piksel boyutunda ve gri tonlamalı olup tek kanal içermektedir. İkincisi ise, ImageNet veri kümesinin bir alt kümesi olan MiniImageNet veri kümesidir. Bu veri kümesi büyük ImageNet veri kümesi kadar büyük ve zor olmasa da en az onun kadar zorlu bir ölçüt olmuştur bu algoritmalar için. MiniImageNet veri kümesi, sınıf başına 600 görüntü olmak üzere toplamda 60000 görüntü içermektedir. Bu görüntüler 84x84 piksel boyutunda ve renkli (RGB) olup üç kanal içermektedir.

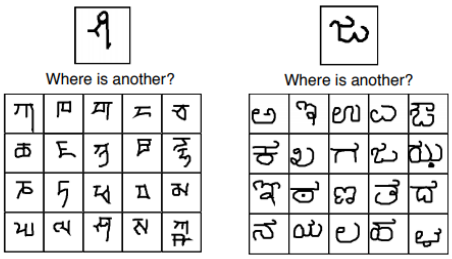
Az örnekle öğrenme problemleri için kullanılan veri kümesi sayısı diğer problemlere oranla çok daha azdır. Az örnekle öğrenme problemleri için kullanılan veri kümelerinin her biri bu bölümde detaylı olarak anlatılacaktır.

2.1.1. Omniglot

Omniglot veri kümesi (Lake vd., 2011), 50 alfabeden yaklaşık 1623 karakter ve her bir karakter için 20 adet görüntü içermektedir. Veri kümesindeki her bir karakter 20 farklı kişi tarafından çizilmiştir. Bu görüntüler 28x28 piksel boyutunda ve gri tonlamalı olup tek kanala sahiptir. Bu veri kümesi, Şekil 2.1’ de gösterildiği gibi Latin ve Kore dilleri gibi yaygın dillerin yanında çok daha az bilinen yerel diller, hatta hayali üretilen Aurek-Besh ve Klingon dillerinden karakterler içermektedir

Şekil 2.1: Omniglot veri kümesinden 525 örnek karakter (Kaynak: Lake et al., 2011).

(Lake vd., 2019) tek seferde öğrenme problemini bu veri kümesiyle oluşturdukları çok sayıda 20-yönlü karakter sınıflandırma problemi için incelemişlerdir. Yaptıkları deneylerde deneklere Şekil 2.2' de gösterildiği gibi yeni bir karakter gösterilmiş ve rastgele olarak üretilen aynı alfabedeki 20 farklı karakterden gösterilen karakter ile aynı karakterin bulunması istenmiştir. Veri kümesindeki 50 alfabenin 30 alfabesi eğitim, 20 alfabesi ise test olarak ayrılmıştır.

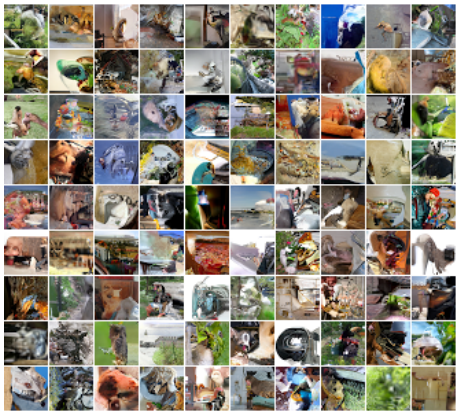


Şekil 2.2: Tek seferde öğrenme problemi için 2 örnek deneme (Kaynak: Lake et al., 2019).

2.1.2. MiniImageNet

MiniImageNet veri kümesi (Vinyals vd., 2016), 100 sınıfının her biri için 600 adet görüntünün rastgele olarak ILSVRC-12 veri kümesinden (Russakovsky vd., 2015) seçildiği ve daha büyük boyutta olan ILSVRC-12 veri kümesinin değiştirilmiş bir sürümüdür veya alt kümesidir. Çok fazla sayıda örnekten oluşan ImageNet veri kümesinin gerektirdiği yüksek hesaplama ve hafıza dezavantajlarına karşı bu veri

kümesini oluşturmuşlardır. Literatürde ağırlıklı olarak (Ravi & Larochelle, 2017) tarafından belirlenen sınıf ayrımları kullanılmaktadır. Bu ayrımda eğitim için 64, doğrulama için 16 ve test için 20 sınıf kullanılmaktadır. Tüm görüntüler 84x84 piksel boyutundadır.



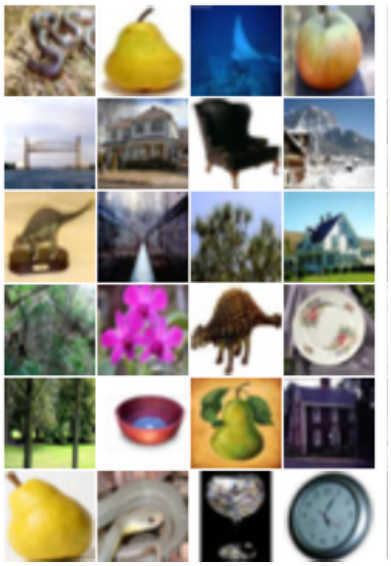
Şekil 2.3: MiniImageNet veri kümesinden örnek bir kesit.

2.1.3. Fewshot-CIFAR100

Literatürde genellikle FC100 olarak kısaltılmış ismiyle geçmektedir. Bu veri kümesinin oluşturulma amacı, yazarları tarafından MiniImageNet üzerinde alınan başarılı sonuçların başka veri kümeleri için de geçerli olduğunu gösterebilmek olarak açıklanmıştır (Oreshkin vd., 2018). Yani sonuçların MiniImageNet''in ötesinde genelleştirildiğini doğrulamak için oluşturulmuş bir veri kümesidir. Ayrıca bu veri kümesindeki görüntüler daha küçük boyutlu olduğu için az örnekle öğrenme problemleri için daha zorlu bir veri kümesidir. Yazarların önerdikleri sınıf ayrımı sayesinde bu sınıflar arasında bilgi örtüşmesini en aza indirilmiştir ve bu da oluşturulan FC100 veri kümesinin Omniglot gibi veri kümelerinden daha zorlu olduğunun bir başka göstergesidir.

Orijinal CIFAR100 (Krizhevsky, 2009) veri kümesi 100 farklı sınıfa ait 32x32 piksel boyutunda renkli görüntülerden oluşmaktadır ve sınıf başına 600 görüntü içermektedir. FC100 veri kümesi için, orijinal CIFAR100 veri kümesinde bulunan 100 sınıf ayrıca 20 üst sınıfa ayrılmıştır ki veri örtüşmesi en aza indirilebilsin ve daha zorlu bir veri kümesi elde edilebilsin. Böylece, eğitim bölümü

12 üst sınıfa ait toplamda 60 sınıf içerirken, doğrulama ve test her biri 5 üst sınıfa ait 20 sınıf içermektedir.



Şekil 2.4: FC100 veri kümesinden örnek bir kesit.

2.1.4. Caltech-UCSD Birds

Caltech-UCSD Birds (CUB olarak isimlendirilecek) veri kümesi eğitim için 100, doğrulama için 50 ve test için 50 sınıf olarak toplamda 200 sınıftan oluşan kuş görüntülerini içermektedir (Wah vd., 2011). Bu veri kümesinde toplamda 11788 kuş görüntüsü bulunmaktadır ve her bir görüntü 84x84 piksel boyutuna sahiptir.

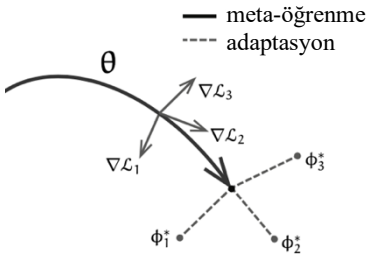
Şekil 2.5: CUB veri kümesinden örnek bir kesit.

2.2. DERİN ÖĞRENME TEMELLİ META-ÖĞRENME YÖNTEMLERİ

Bu bölümde, MAML, ProtoNet, Reptile ve MAML++ gibi meta-öğrenme yöntemleri bölümler halinde özetlenecektir. Karşılaştırması yapılacak meta-öğrenme algoritmalarının seçiminde, literatürde önemli ağırlığı bulunan algoritmalar ve yeni olarak sunulan algoritmaların karşılaştırmasında çokça tercih edilmekte olan algoritmalar (MAML ve ProtoNet) dikkate alınmıştır.

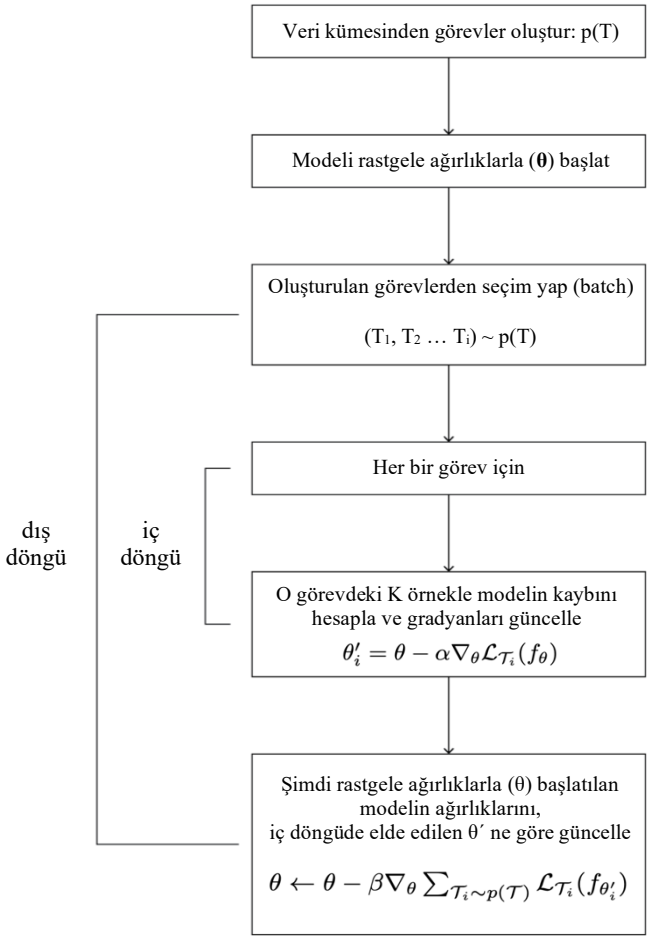
2.2.1. MAML

Model-Agnostic Meta-Learning (MAML), Gradient Descent algoritması ile eğitilmiş herhangi bir model için uyumlu ve çeşitli farklı öğrenme problemlerine (sınıflandırma, regresyon ve pekiştirici öğrenme gibi) uygulanabilir bir algoritma olarak Finn ve arkadaşları tarafından 2017 yılında önerildi (Finn vd., 2017). Meta-öğrenmenin olmadığı durumlarda, Gradient Descent algoritması ile eğitilecek olan model rastgele ağırlıklarla öğrenme sürecine başlar ve hedefe doğru adım adım ilerler. Bu durumda, hedefe doğru gitmeye hangi noktadan başladığımız rastgele seçildiği için sonuca ulaşmak daha fazla adım gerektirir. Meta-öğrenmenin olduğu durumlarda ise başlangıç noktası daha makul bir nokta olarak seçildiği için hedefe daha az adımda ulaşılır. Başlangıç noktasının daha iyi olduğu varsayımı, yeni görevin eski görevlerle benzerlikler içermesinden gelmektedir. Bu benzerlik ne kadar fazla ise, hedefe varma süresi o kadar kısa sürer. Bu adaptasyon durumu, sonraki örneklere olan uzaklık bakımından Şekil 2.6'da gösterilmiştir. Şekil 2.6'da gösterilen θ modelin parametrelerini temsil ederken siyah kalın çizgi ise meta-öğrenme sürecini temsil etmektedir. Şekil 2.6'da verildiği gibi 3 farklı görevimiz olduğunda, her görev için gradyan adımı atılır (gri çizgiler). Modelin parametrelerinin (θ), görev 1, 2 ve 3'ün tüm 3 optimal parametresine de (Φ_1, Φ_2, Φ_3) yakın olduğunu görebiliriz. İyi parametrelerle başlangıç (θ) durumu sayesinde farklı yeni görevlere hızla adapte olabilmeyi sağlar.



Şekil 2.6: MAML algoritmasının adaptasyon süreci (Kaynak: Finn vd., 2017).

Önerilen algoritma, iki optimizasyon döngüsünden oluşmaktadır; dış döngü meta bilgileri kullanarak uygun bir başlangıç bulmaya çalışırken, iç döngü ise çok az etiketli örnekle başlangıç parametrelerini optimize ederek yeni görevi öğrenmeye çalışır. Bu algoritmanın amacı, benzer görevlere daha iyi genelleyen başlangıç parametreleri bulmaktır. Bu sayede, daha iyi başlangıç parametreleriyle oluşturulan model daha az adımla yeni görevleri daha hızlı bir şekilde öğrenebilir.



Şekil 2.7: MAML algoritmasının akışı.

Şekil 2.7' de bu algoritmanın akışı özetlenmiştir. MAML, Gradient Descent algoritmasını kullanan benzer görevler için adım adım ilerlemeyi olabildiğince çabuk sağlayacak bir dizi ağırlık bulmaya çalışır. Bunun için, Gradient Descent algoritmasını bir adım (veya i adım) çalıştırıp ardından bu adımın gerçek sonuca doğru ne kadar ilerleme kaydettiğine bağlı olarak başlangıç ağırlıklarını günceller.

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

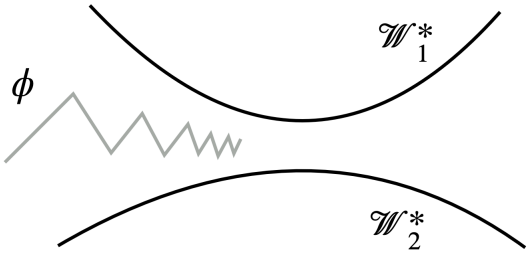
Şekil 2.8: MAML algoritmasının sözde kodu (Kaynak: Finn vd., 2017).

Şekil 2.8' de ise bu algoritmanın orijinal makalesinde paylaşılan sözde kodu verilmiştir. MAML algoritması, satır 1'de belirtildiği gibi θ parametrelerini rastgele seçerek başlar ve sonrasında satır 2 ve 3'de belirtildiği gibi birden fazla görevi içeren ilk grup (batch) ile döngüye (while) başlar. Satır 4 ve 5'te belirtildiği gibi, sıradaki gruptaki her bir görev için, bu göreve ait K adet örneği (k-shot learning) kullanarak modeli eğitir. Ardından, satır 6'da belirtildiği gibi, bu K adet örnek için karşılık gelen yeni test örneklerinde modeli test ederek kaybı (loss) hesaplar ve modelin parametrelerini iyileştirir. Birden fazla görev içeren gruptaki (batch) test hatası, meta-öğrenme sürecinin eğitim hatası olarak kullanılır. Buradaki ana hedef, kaybı en aza indirmektir. Bir sonraki görevleri içeren gruba (batch) geçmeden önce, satır 8'de belirtildiği gibi, modelin (θ) parametreleri Stokastik Gradyan İniş (SGD) kullanılarak güncellenir. Ve bu süreç, bütün görev grupları bitene kadar devam eder.

MAML, gradyanlı iniş ile eğitilmiş herhangi bir model üzerinde çalışabilir ve farklı kayıp fonksiyonları ile kullanılabilir.

2.2.2. Reptile

MAML algoritmasına birçok açıdan benzemekle birlikte daha basit bir varyasyonu olarak ortaya çıkmıştır. Bunun sebebi olarak ise Reptile algoritmasının, MAML algoritmasının tersine, bir sonraki adımı yaklaşık olarak hesaplamasıdır. Bu yaklaşık hesaplamanın temelinde ise şu düşünce vardır; her göreve muhtemelen birden fazla uygun olabilecek ağırlık $\{w_i^*\}$ vardır ve buradaki hedef ise her görev için bu ağırlıklardan $\{w_i^*\}$ en az birine yakın bir başlangıç ağırlıkları $\{W_{init}\}$ bulmaktır. Şekil 2.9'da bu düşüncenin temsili bir gösterimi verilmiştir.



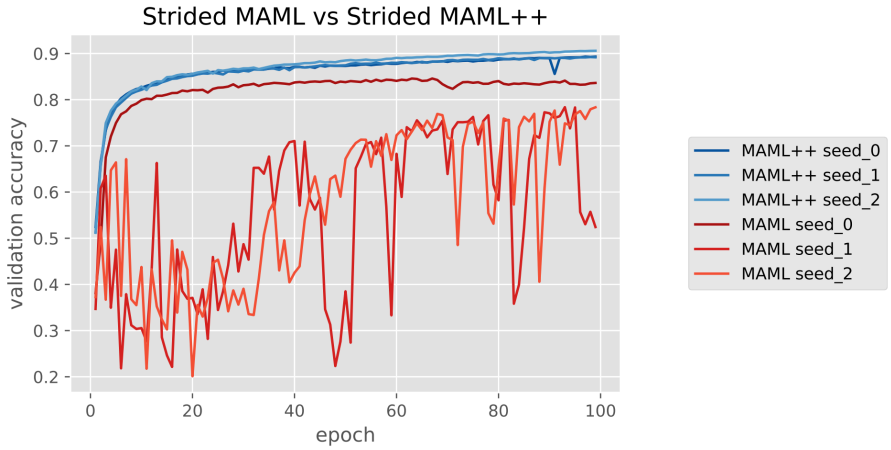
Şekil 2.9: Reptile algoritmasının yaklaşımının gösterimi (Kaynak: Nichol vd., 2018).

Nichol ve arkadaşları, Omniglot ve MiniImageNet veri kümelerini kullanarak yaptıkları deneylerde, Reptile ve MAML algoritmalarının az öğrenme problemleri için benzer sonuçlar sağladığını gösterdiler (Nichol vd., 2018). Ayrıca Reptile algoritmasının çözüme daha hızlı bir şekilde yakınsadığını belirtip bunun sebebinin ise daha düşük sapmalara sahip olması olduğunu gösterdiler.

2.2.3. MAML++

MAML++ algoritması, kullanılan sinir ağı mimarisine ve hiperparametrelere bağlı olarak MAML algoritmasının eğitim sürecinin kararsız bir şekilde ilerleyebileceğini göstererek bu duruma bir düzenleme getirebilmek üzerine Antoniou ve arkadaşları tarafından 2018 yılında önerildi (Antoniou vd., 2018). Orijinal MAML, birden fazla iç adım olsa bile dış döngü güncelleme işlemi için sadece son adımdaki ağırlıkları kullandığından, bu durumun kurulan yapıyı eğitim sürecinde hassas bir duruma getirdiği ve gradyan bozulmalarına sebebiyet verdiği gösterilmiştir. Çok Adımlı Kayıp Optimizasyonu (MSL), iç döngünün son adımı yerine her adımdan sonra ağırlıklı kayıp toplamını hesaplamak için önerilmektedir.

Ağırlıklı yöntem, modelin önceki adımlardan ziyade sonraki adımlara da önem vermesini sağlar ve bu durum eğitim sürecinin kararsızlığını gidermek için kullanılır.



Şekil 2.10: Eğitim sürecindeki istikrarsızlık (Kaynak: Antoniou vd., 2018).

Şekil 2.10'da orijinal makaleden alınan sonuçlar gösterilmiştir. Çözüm olarak ise sadece son ağırlıklara göre güncelleme yapmak yerine bütün iç adımlar dikkate alınarak elde edilen ortalama üzerinden dış döngü güncellemesi yapılmıştır. Bu sayede eğitim sürecindeki istikrarsızlığı ortadan kaldırdıklarını yaptıkları deneylerle göstermişlerdir ve MAML algoritması üzerine yaptıkları bu geliştirme ile daha stabil bir eğitim süreci elde edilmesini sağlamışlardır.

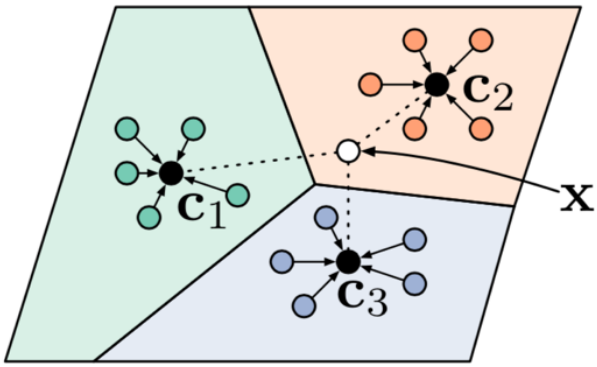
2.2.4. ProtoNet

Prototypical Networks (ProtoNet), her sınıf için bir prototipin oluşturulduğu ve her sınıfın prototip temsillerine olan uzaklığını hesaplayarak sınıflandırma işleminin gerçekleştirilebildiği bir algoritma olarak Snell ve arkadaşları tarafından 2017 yılında önerildi (Snell vd., 2017). ProtoNet, her bir 'k' sınıfına ait olabilecek M boyutlu özellik vektörü (c_k) veya bir başka deyişle bir prototip bulmaya çalışır. Her bir sınıf için oluşturulacak olan prototip vektörünü kodlamak için bir f_θ fonksiyonu kullanır. Her bir prototip (c_k), o sınıfa ait olan veri örneklerinin ortalama vektörü olarak tanımlanır.

Sonrasında ise, d olarak adlandırılan bir uzaklık hesabı fonksiyonunu kullanarak, genellikle Öklid, yeni gelen bir veri için bütün sınıflar üzerindeki dağılıma bakarak hangi sınıfa ait olduğunu bulmaya çalışır. Öğrenme süreci ise bu

doğru sınıfın atanması işleminden doğacak kaybın SGD kullanılarak en aza indirilmeye çalışılması olarak tanımlanmıştır.

Bu algoritmanın çıkış fikri ise sinirsel bir ağ tarafından öğrenilen bir temsil alanında her sınıfın kendi örnekleriyle (prototip) temsil edilebileceği düşüncesidir. Böyle bir yaklaşımın, diğer meta-öğrenme yaklaşımlarından çok daha basit olmasına karşın onlarla benzer sonuçlar aldığı ayrıca yapılan çalışmalarda gösterilmiştir. Temsili olarak prototiplerin oluşturulması Şekil 2.11'de gösterilmiştir. Şekil 2.11'de şimdiye kadar alınan örneklerden oluşturulan 3 prototip gözükmektedir (C_1, C_2 ve C_3) ve bu prototipler alınan örneklerin ortalaması kullanılarak hesaplanmıştır. Yeni gelen X örneği ise seçilmiş olan yakınlık fonksiyonuna göre (Öklid fonksiyonu gibi) en yakın olduğu prototipe (C_1, C_2 veya C_3) atanır ve sonrasında ise prototipler tekrardan hesaplanır.

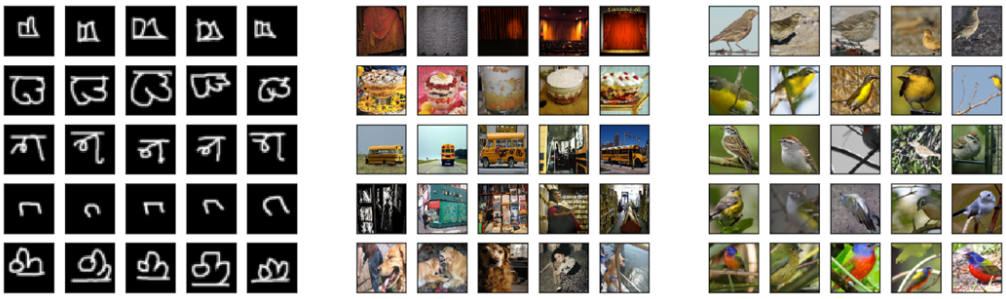


Şekil 2.11: ProtoNet algoritmasının prototipler oluşturmasının temsili gösterimi.

3. DENEYLER

Bu bölümün ilk kısmında kurulan deney ortamı anlatılacak, sonrasında ise meta-öğrenme algoritmalarının (MAML ve ProtoNet) az örnekle öğrenme deneylerinde aldıkları sonuçlar farklı veri setleri için başlıklar altında ayrı ayrı verilecektir. Alınan deney sonuçları ise bir sonraki bölümde değerlendirilecektir ve MAML, ProtoNet gibi meta-öğrenme yaklaşımlarının az örnekle öğrenme problemlerindeki başarısının hangi parametrelere bağlı olduğu gösterilecektir.

Bahsedilen yaklaşımların farklı veri kümeleri üzerinde adil bir şekilde karşılaştırılabilmesi için oluşturulacak olan ortamın bütün deneyler için aynı altyapıyı sunması gerekmektedir. Literatürde öne sürülmüş olan her algoritmanın farklı veri yükleme alternatiflerini kullanıyor olması, Chen ve arkadaşları tarafından yapılmış olan bir diğer çalışmada da belirtilerek bu durumun algoritmaların adil bir şekilde karşılaştırılabilmesini zorlaştırdığı belirtilmiştir (Chen vd., 2019). Bu sebepten ötürü, veri katmanı için esnek bir yapıya sahip olan Torchmeta kütüphanesi tercih edilmiştir (Deleu vd., 2019). Bu sayede bütün algoritmaların veri yükleme işlemleri sabitlenerek kullanılacak algoritmadan bağımsız olarak çalışması sağlanmıştır. Torchmeta, PyTorch kullanılarak yazılmış, az örnekle öğrenme ve meta-öğrenme için kullanıma sunulmuş açık kaynaklı bir veri yükleyici (dataloader) koleksiyonudur. Bu veri yükleyici koleksiyon, Omniglot ve MiniImageNet gibi çok bilindik veri kümelerini içerisinde barındırmakla beraber ayrıca farklı veri kümelerini de yükleyebilmek için gerekli ortamı sağlamaktadır. Bu nedenle veri yükleme katmanı olarak seçilmiştir.



Şekil 3.1: 5-yönlü 5-örnek veri kümesi örnekleri; Omniglot (sol), MiniImageNet (orta) ve Caltech-UCSD Birds (sağ).

Vinyals ve arkadaşları tarafından az örnekle öğrenme için önerilmiş olan deney düzenekleri oluşturulmuştur (Vinyals vd., 2016). 1-shot ve 5-shot yapılandırmaları kurularak modele sadece 1 veya 5 örnek gösterilerek N-way sınıflandırmayı hızlı bir şekilde yapması istenmiştir. N-yönlü sınıflandırma problemi şu şekilde ayarlanır: N adet daha önceden modele verilmemiş olan sınıf seçilir, sonrasında ise modele N sınıfın her biri için olmak üzere K farklı veri örneği verilir ve modelin bu N adet sınıfı sınıflandırma yeteneği doğruluk bakımından değerlendirilir. Şekil 3.1'de, üç farklı veri kümesinden, modele yukarıda bahsettiğimiz yapılandırmaya uygun şekilde verilecek olan veri kümesi örnekleri bulunmaktadır. En solda bulunan Omniglot örneğini ele alırsak eğer, eğitim sürecinde modele bu şekilde birden fazla veri kümesi örneği verilecektir. Bu küçük veri kümesi örneklerinin her biri görev (task) olarak isimlendirilmektedir ve algoritmaların asıl amacı farklı sınıflar içeren görevler için daha genel bir yapılandırma oluşturmak ve sonuç olarak sınıflandırma yeteneklerini genelleştirebilmektir.

3.1. DENEY ORTAMI

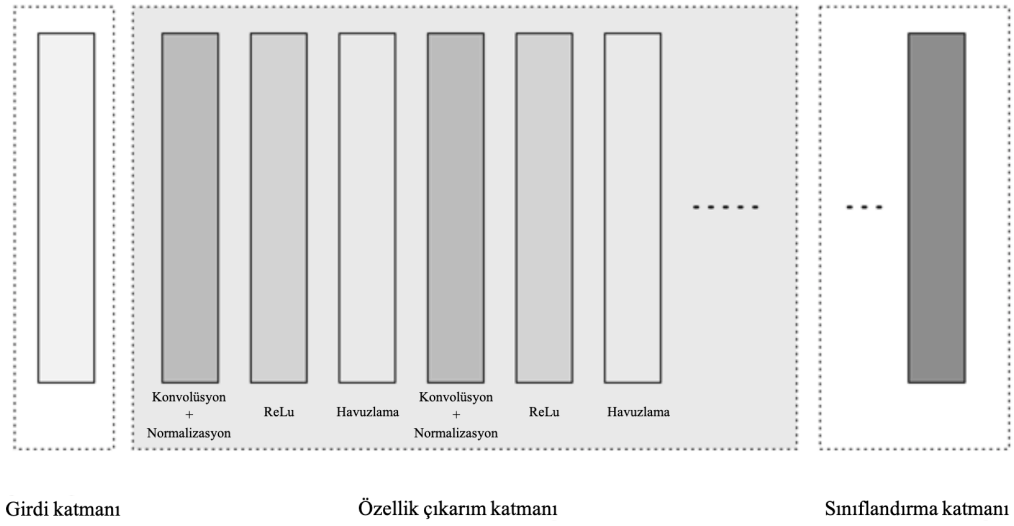
Omniglot, MiniImageNet, FC100 ve CUB veri kümeleri üzerinde meta-öğrenme algoritmalarının test edilip farklı parametreler için alınan sonuçların karşılaştırılması hedeflenmektedir. Ayrıca bu algoritmaların başlangıç durumları da birkaç örnek üzerinden gösterilmek üzere eğitilen ağırlıklarla (pre-train) başlatılarak test edilecektir. Yani başlangıç ağırlıkları rastgele ve önceden eğitilmiş olmak üzere 2 farklı şekilde eğitim işlemleri gerçekleştirilecektir.

Rastgele başlangıç ağırlıkları kullanılırken rastgelelik durumlarının bütün algoritmalarda aynı şekilde işlemesi için rastgele sayı üretilirken kullanılan "seed" değeri bütün deneylerde sabitlenmiştir. Böylece bütün deneyler için geçerli olacak deterministik bir karşılaştırma ortamı oluşturulmuştur.

Önceden eğitilmiş ağırlıkların oluşturulması için ise Tiny ImageNet veri kümesi tercih edilmiştir (Le & Yang, y.y.). Tiny ImageNet veri kümesinde 200 sınıfa bölünmüş toplamda 120000 görüntü bulunmaktadır. Her sınıfta ise 500 eğitim görüntüsü, 50 doğrulama görüntüsü ve 50 test görüntüsü bulunmaktadır. Bu veri

kümesi büyük ImageNet veri kümesinin küçük bir alt kümesidir. Önceden eğitilmiş ağırlıkların oluşturulmasında bu veri kümesinin (Tiny ImageNet) tercih edilme sebebi ise hem zaman hem de hesaplama bakımından elimizde bulunan kısıtlı donanımlardır.

CNN mimarisi (backbone) olarak ise daha önce benzer çalışmalarda da sıkça kullanılmakta olan bir yapı tercih edilmiştir (Vinyals vd., 2016). Bu yapı içerisinde 4 adet konvolüsyonel blok içermektedir. Her bir blok içerişinde ise 64 filtreli 3x3 konvolüsyon, ardından normalizasyon (Ioffe & Szegedy, 2015), doğrusal olmayan ReLu aktivasyonu ve 2x2 havuz katmanları bulunmaktadır. Sonrasında ise sınıflandırma katmanları gelmektedir. Bütün modellerde optimizasyon yöntemi olarak ise Adam tercih edilmiştir (10^{-3} öğrenme oranı ile). Yapının daha anlaşılır olması için, yapı içerisinde bloklar Şekil 3.2' de görsel hale getirilmiştir.



Şekil 3.2: Kullanılan CNN Mimarisi.

Az örnekle öğrenme deneylerini MAML algoritması ile çalıştırmak için gereken bütün parametreler belirlenmiştir. Bu parametrelerin belirlenmesinde literatürdeki çalışmalarda sıklıkla kullanılan yapılandırma ayarları tercih edilmiştir. Tablo 3.1'de MAML algoritmasının az örnekle öğrenme problemlerinde test edilmesi için belirlenmiş olan bütün parametreler toplu bir şekilde verilmiştir.

Tablo 3.1: MAML algoritması için seçilen parametre değerleri.

Parametreler	Seçilen değerler
Sınıf sayısı (way)	5 ve 10
Örnek sayısı (shot)	1 ve 5
Eğitimdeki adım sayısı	1 ve 5
Testteki adım sayısı	5 ve 10
Adım genişliği	0,1 ve 0,2 ve 0,4
Yığın boyutu (batch)	8
Öğrenme oranı (Adam)	10^{-3}

Tablo 3.2'de ise ProtoNet algoritmasının az örnekle öğrenme problemlerinde test edilmesi için belirlenmiş olan bütün parametreler verilmiştir.

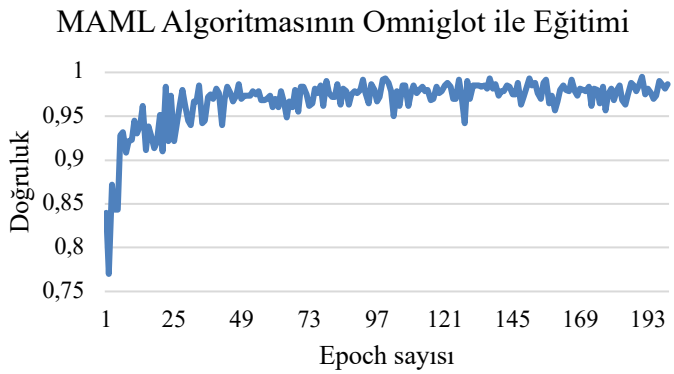
Tablo 3.2: ProtoNet algoritması için seçilen parametre değerleri.

Parametreler	Seçilen değerler
Sınıf sayısı (way)	5 ve 10
Örnek sayısı (shot)	1 ve 5
Yığın boyutu (batch)	8
Öğrenme oranı (Adam)	10^{-3}

3.2. MAML SONUÇLARI

200 epoch boyunca MAML algoritması eğitilerek model oluşturulmuştur. Bu epoch sayısı belirlenirken, bütün veri kümesinin görünür olmasına dikkat edilerek olabildiğince küçük seçilmiştir. Örneğin MiniImageNet veri kümesinde eğitim için

64 sınıf bulunmakta ve her sınıf 600 görüntü içermektedir. Yani bu veri kümesinde eğitim amacıyla ayrılmış toplamda 64*600 = 38400 adet görüntü bulunmaktadır. Örnek olarak 5-yönlü 5-örnek olarak kurulacak olan bir yapılandırmaya bakarsak eğer; 1 epoch içerisinde 5*5*(yığın boyutu) kadar görüntü bulunmaktadır. Yığın boyutunun 8 alındığını düşünürsek eğer toplamda 200 görüntü 1 epoch içerisinde işlenmektedir. 200 epoch olduğunda ise toplamda işlenen görüntü sayısı 40000 olacaktır. Ayrıca Şekil 3.3'de, 5-yönlü 5-örnek yapılandırmasının Omniglot veri kümesi üzerindeki eğitim sürecine ait doğruluk oranlarını içeren gerçek verilerden oluşan bir gidişat bulunmaktadır. Bu gidişat seçilen epoch sayısının yeterli olduğunu göstermektedir.



Şekil 3.3: MAML algoritmasının Omniglot veri kümesiyle eğitim sürecinin gidişatı.

3.2.1. Omniglot Veri Kümesi Üzerindeki Sonuçları

Omniglot veri kümesi için, rastgele ağırlıklarla MAML algoritması başlatıldığında elde edilen doğruluk (accuracy) oranları Tablo 3.3'de verilmiştir. Her bir deney için 3 farklı adım genişliği ve ayrıca 4 farklı adım sayısı yapılandırması test edilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.3: MAML algoritmasının Omniglot veri kümesi doğruluk oranları.

Adım sayıları	**Adım genişliği**	5-yönlü		10-yönlü	
		1-örnek	5- örnek	1- örnek	5- örnek
Eğitim: 1 Test: 5	0,1	**87,77 ± 0,22**	**96,96 ± 0,08**	**79,47 ± 0,20**	**93,84 ± 0,09**
	0,2	79,01 ± 0,31	94,20 ± 0,14	68,80 ± 0,23	84,90 ± 0,05
	0,4	60,58 ± 0,16	81,45 ± 0,14	61,79 ± 0,12	76,57 ± 0,09
Eğitim: 1 Test: 10	0,1	**87,93 ± 0,23**	**97,25 ± 0,07**	**80,35 ± 0,20**	**94,97 ± 0,06**
	0,2	82,01 ± 0,29	96,01 ± 0,09	73,70 ± 0,21	86,29 ± 0,23
	0,4	63,55 ± 0,18	91,11 ± 0,10	67,84 ± 0,11	82,63 ± 0,11
Eğitim: 5 Test: 5	0,1	**86,95 ± 0,22**	**97,09 ± 0,07**	**81,65 ± 0,19**	**94,80 ± 0,09**
	0,2	83,99 ± 0,25	96,88 ± 0,04	77,59 ± 0,23	86,55 ± 0,10
	0,4	69,49 ± 0,16	78,52 ± 0,14	69,83 ± 0,11	77,60 ± 0,13
Eğitim: 5 Test: 10	0,1	**87,96 ± 0,22**	**97,42 ± 0,07**	**82,97 ± 0,18**	**96,50 ± 0,05**
	0,2	85,99 ± 0,23	97,39 ± 0,03	80,43 ± 0,19	95,68 ± 0,03
	0,4	77,21 ± 0,15	89,15 ± 0,09	76,09 ± 0,09	87,16 ± 0,06

Orijinal çalışmada 5-yönlü 1-örnek için 98,70 ± 0,4 doğruluk sonucu elde edilirken 5-yönlü 5-örnek için ise 99,9 ± 0,1 doğruluk sonucu elde edilmiştir. Rastgele olarak başlatılan MAML algoritması, Omniglot veri kümesi üzerinde 5-yönlü 5-örnek için beklenen sonuca %2'li bir sapma ile yaklaşık olarak %98 gibi bir sonuç almasına rağmen örnek sayısının daha az olduğu 5-yönlü 1-örnek için olması gerekenden yaklaşık %10 daha az olan %87 gibi bir sonuca ulaşmaktadır. Bu sebepten ötürü, 5-yönlü 1-örnek deneyi, adım genişliği 0,1 olmak üzere, 1000 epoch çalıştırılarak olması beklenilen sonuca ulaşılabilmiştir. 1000 epoch sonucunda elde edilen model aynı test düzeneği ile test edildiğinde ise %95 güven aralığında 94,07 ± 0,15 doğruluk sonucunu vermiştir. Bu sonuçta olması gereken sonuçtan yaklaşık olarak %4 daha azdır ve bu fark kabul edilebilir bir fark olmakla birlikte MAML algoritmasının PyTorch üzerinde doğru bir şekilde çalıştırılabildiğini göstermektedir.

Yapılan deneyler sonucunda, MAML algoritması için adım genişliğinin 0,2 veya 0,4 değerleri yerine 0,1 olarak seçilmesinin bütün deney varyasyonlarında en iyi sonuçları verdiği gösterilmiştir. Özellikle adım genişliği 0,4 olduğunda aradaki fark açıkça gözükmektedir ve daha kötü sonuçlar elde edilmektedir. Örnek olarak SGD'nin eğitimde 1 adım testte ise 5 adım çalıştırıldığı, yani öğrenmenin daha da zorlaştırıldığı durumda, 5-yönlü 1-örnek için yapılan deney sonuçlarını içeren sütuna

bakarsak eğer, adım genişliği 0,4 iken elde edilen doğruluk 60,58 olurken adım genişliği 0,1 olarak alındığında doğruluk 87,77 gibi büyük bir yükseliş göstermiştir. Aralarında yaklaşık %27'lik bir fark bulunmaktadır ve bu farkın tek sebebi adım genişliğinin 0,4 yerine 0,1 olarak alınmış olmasıdır.

Hem eğitim hem de test süreci içerisinde SGD farklı adım sayıları ile test edilerek sonuçları yukarıda verilen tablolarda ayrı ayrı gösterilmiştir. Eğitim sırasında adım sayısı olarak 1 ve 5 değerleri tercih edilirken test sürecinde ise 5 ve 10 değerleri tercih edilmiştir. Eğer 1 adım eğitimde 5 adım testte kullanırsak, SGD eğitim sürecinde gelen görüntüyü tek bir adım kullanarak öğrenmeye çalışırken test sürecinde ise 5 adım kullanarak gelen yeri görüntüden çıkarım yapmaya çalışmaktadır.

Eğitim için 1 adım tercih edildiğinde test sürecinde 5 adım yerine 10 adım kullanmanın sonuçları arttırdığı gözlemlenmektedir. Aynı durum eğitim için 5 adım kullanıldığı zaman da geçerlidir. Bu olumlu yansımanın yüzdesi ise tercih edilen adım genişliğine göre farklılıklar göstermekle birlikte daha büyük adım genişliğinin (0,2 veya 0,4) tercih edildiği deneylerde olumlu yansıma da daha fazla gözükmektedir.

5-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen görüntü sayısı arttırıldığında doğruluk oranı da yükseltmektedir. Adım genişliğinin 0,1 olarak alındığı ilgili satırlara bakarsak eğer aradaki farkın bütün tablolarda yaklaşık olarak %10 olduğunu gözlemleyebiliriz. Adım genişliğinin 0,1 değerinden farklı olarak seçildiği durumlarda ise bu artışın yüzdesinin daha fazla olduğu gözlemlenmektedir.

Omniglot veri kümesi tek kanal içeren görüntülerden oluştuğundan ve literatürdeki çalışmalarda başlangıç ağırlıklarının üç kanal içeren görüntülerden oluşan ImageNet olarak kullanıldığı durumlarda sonuçların daha kötüye gittiği gösterilmiştir. Bu sebeplerden dolayı Omniglot veri kümesi için sadece rastgele ağırlıklarla MAML algoritması çalıştırılmıştır. Daha önceden eğitilmiş bir modelin ağırlıkları kullanılarak MAML algoritması çalıştırılmamıştır. Tablo 3.4'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deneylerin sonuçları ile orijinal makalesindeki (Finn vd., 2017) paylaşılmış olan deneylerin sonuçlarının

karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.4: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

Omniglot	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	98,70 ± 0,4	99,9 ± 0,1
Kurulan deney düzeneği	94,07 ± 0,15	97,42 ± 0,07

3.2.2. MiniImageNet Veri Kümesi Üzerindeki Sonuçları

MiniImageNet veri kümesi için, rastgele ağırlıklarla MAML algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.5'de verilmiştir. Her bir deney için 3 farklı adım genişliği ve ayrıca 4 farklı adım sayısı yapılandırması test edilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.5: MAML algoritmasının MiniImageNet veri kümesi doğruluk oranları.

Adım sayıları	**Adım genişliği**	5-yönlü		10-yönlü	
		1-örnek	5- örnek	1- örnek	5- örnek
Eğitim: 1 Test: 5	0,1	**32,91 ± 0,25**	**48,01 ± 0,24**	21,11 ± 0,14	**32,99 ± 0,15**
	0,2	31,34 ± 0,24	46,70 ± 0,26	**21,23 ± 0,14**	32,94 ± 0,15
	0,4	30,80 ± 0,23	42,52 ± 0,24	19,45 ± 0,13	28,97 ± 0,14
Eğitim: 1 Test: 10	0,1	**33,34 ± 0,24**	**49,09 ± 0,25**	**21,64 ± 0,14**	**33,24 ± 0,16**
	0,2	31,47 ± 0,24	46,80 ± 0,25	21,29 ± 0,13	32,91 ± 0,15
	0,4	30,92 ± 0,24	42,63 ± 0,25	19,96 ± 0,13	29,52 ± 0,15
Eğitim: 5 Test: 5	0,1	**33,29 ± 0,23**	**50,89 ± 0,25**	**21,83 ± 0,14**	**35,48 ± 0,16**
	0,2	31,54 ± 0,24	48,06 ± 0,25	21,53 ± 0,13	34,65 ± 0,15
	0,4	28,91 ± 0,23	46,51 ± 0,26	20,12 ± 0,13	32,85 ± 0,15
Eğitim: 5 Test: 10	0,1	**34,22 ± 0,24**	**50,69 ± 0,25**	**22,98 ± 0,14**	**36,21 ± 0,16**
	0,2	32,33 ± 0,24	48,78 ± 0,26	22,48 ± 0,13	35,32 ± 0,16
	0,4	30,49 ± 0,22	47,18 ± 0,27	21,12 ± 0,13	33,54 ± 0,15

Rastgele ağırlıklarla başlatmanın önceden eğitilmiş modelin ağırlıklarını kullanarak başlamayla arasındaki farkı görebilmek için bazı deneyler eğitilmiş ağırlıklarla başlayacak şekilde ayarlanıp tekrar edilmiştir. Eğitilmiş ağırlıklar ise, aynı yapı kullanılarak klasik model eğitimi düzeneğini kurarak (meta-learning eğitim sürecinden farklı olarak), Tiny ImageNet veri kümesi üzerinden modelin eğitilmesi ile elde edilmiştir.

Tablo 3.6: Eğitilmiş ağırlıkla başlangıç için alınan doğruluk oranları.

MiniImagenet **Eğitilmiş ağırlıklarla başlangıç**	5-yönlü	
	1-örnek	5-örnek
Adım genişliği 0,1	45,90 ± 0,31	59,10 ± 0,26

Tiny ImageNet veri kümesiyle eğitilmiş modelin ağırlıklarının başlangıç durumu olarak kullanılması orijinal çalışma sonuçlarına yakın değerler elde edilmesini sağlamıştır ve bu sonuçlar Tablo 3.6'da paylaşılmıştır. Orijinal çalışmada başlangıç durumu olarak büyük ImageNet veri kümesi üzerinden elde edilmiş olan ağırlıkların kullanılması sonucunda 5-yönlü 1-örnek için 48,70 ± 1,84 doğruluk sonucu elde edilirken 5-yönlü 5-örnek için ise 63,11 ± 0,92 doğruluk sonucu elde edilmiştir. Aradaki %3'lük farkın bu sebepten ötürü kabul edilebilir bir fark olduğu gözükmektedir.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine eğitilmiş ağırlıklar ile modelin başlatılması doğruluk oranlarında ciddi artışlara yol açmıştır. 5-yönlü 1-örnek için %11'lik bir artış gösterirken 5-yönlü 5-örnek için ise yaklaşık %9'luk bir artış göstermiştir.

Yapılan deneyler sonucunda, Omniglot veri kümesinde olduğu gibi MiniImageNet veri kümesi üzerinde de MAML algoritması için adım genişliğinin 0,2 veya 0,4 değerleri yerine 0,1 olarak seçilmesinin bir deney hariç bütün deney varyasyonlarında en iyi sonuçları verdiği gösterilmiştir. 10-yönlü 1-örnek deneyi için adım genişliği 0,2 olduğunda %0,12 gibi küçük bir farkla da olsa adım genişliği 0,1 olarak alınan duruma göre daha iyi bir sonuç alınmıştır. Bu veri kümesi üzerinde farklı adım genişliği için alınan sonuçlar daha yakın olmuştur. Eğer 5-yönlü deneyler için (1-örnek ve 5-örnek) bakacak olursak %4 ila %7 arasında değişiklik

göstermektedir. Aynı durum Omniglot veri kümesinde %8 ila %27 arasında değişiklik göstermekteydi.

Eğitim için 1 adım tercih edildiğinde test sürecinde 5 adım yerine 10 adım kullanmanın Omniglot veri kümesi üzerinde olduğu gibi MiniImageNet veri kümesi üzerinde alınan sonuçları da arttırdığı gözlemlenmektedir. Aynı durum eğitim için 5 adım kullanıldığı zaman da geçerlidir. Ama bu artış genellikle %1 seviyelerinde kalmış, daha fazla bir iyileştirme elde edilememiştir. Bu olumlu yansımanın yüzdesi ise tercih edilen adım genişliğine göre farklılıklar göstermekle birlikte daha büyük adım genişliğinin (0,2 veya 0,4) tercih edildiği deneylerde olumlu yansıma da yaklaşık %7'lere çıkarak daha fazla gözükmektedir.

5-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen görüntü sayısının arttırılması doğruluk oranını da yükseltmektedir. Adım genişliğinin 0,1 olarak alındığı ilgili satırlara bakarsak eğer aradaki farkın bütün tablolarda yaklaşık olarak %15 olduğunu gözlemleyebiliriz. Tablo 3.7'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deneylerin sonuçları ile orijinal makalesindeki (Finn vd., 2017) paylaşılmış olan deneylerin sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.7: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

MiniImageNet	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	48,70 ± 1,84	63,11 ± 0,92
Kurulan deney düzeneği	45,90 ± 0,31	59,10 ± 0,26

3.2.3. Fewshot-CIFAR100 Veri Kümesi Üzerindeki Sonuçları

Fewshot-CIFAR100 (FC100 olarak bahsedilecek) veri kümesi için, rastgele ağırlıklarla MAML algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.8'de verilmiştir. Her bir deney için 3 farklı adım genişliği ve ayrıca 4 farklı adım sayısı yapılandırması test edilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.8: MAML algoritmasının FC100 veri kümesi doğruluk oranları.

Adım sayıları	**Adım genişliği**	5-yönlü		10-yönlü	
		1-örnek	5- örnek	1- örnek	5- örnek
Eğitim: 1 Test: 5	0,1	**40,90 ± 0,31**	**56,17 ± 0,28**	**27,58 ± 0,17**	**41,34 ± 0,18**
	0,2	40,26 ± 0,30	54,41 ± 0,29	27,34 ± 0,18	38,32 ± 0,17
	0,4	35,59 ± 0,28	49,38 ± 0,28	23,77 ± 0,14	33,27 ± 0,15
Eğitim: 1 Test: 10	0,1	**40,89 ± 0,30**	**56,69 ± 0,29**	**28,07 ± 0,17**	**42,41 ± 0,18**
	0,2	40,08 ± 0,29	54,33 ± 0,29	27,59 ± 0,18	38,89 ± 0,18
	0,4	35,77 ± 0,28	49,45 ± 0,28	24,18 ± 0,16	34,97 ± 0,17
Eğitim: 5 Test: 5	0,1	**41,67 ± 0,30**	**58,73 ± 0,29**	28,90 ± 0,17	**44,98 ± 0,17**
	0,2	35,10 ± 0,26	58,06 ± 0,28	**29,32 ± 0,17**	43,30 ± 0,18
	0,4	31,08 ± 0,24	52,40 ± 0,28	21,08 ± 0,14	37,13 ± 0,15
Eğitim: 5 Test: 10	0,1	**42,65 ± 0,31**	**59,49 ± 0,29**	**29,58 ± 0,18**	**46,02 ± 0,17**
	0,2	38,47 ± 0,29	58,49 ± 0,29	29,54 ± 0,17	44,42 ± 0,18
	0,4	34,20 ± 0,28	52,06 ± 0,30	21,47 ± 0,16	38,65 ± 0,18

Rastgele ağırlıklarla başlatmanın önceden eğitilmiş modelin ağırlıklarını kullanarak başlamayla arasındaki farkı görebilmek için bazı deneyler eğitilmiş ağırlıklarla başlayacak şekilde ayarlanıp tekrar edilmiştir. Eğitilmiş ağırlıklar ise, aynı yapı kullanılarak klasik model eğitimi düzeneğini kurarak (meta-learning eğitim sürecinden farklı olarak), Tiny ImageNet veri kümesi üzerinden modelin eğitilmesi ile elde edilmiştir.

Tablo 3.9: Eğitilmiş ağırlıkla başlangıç için alınan doğruluk oranları.

FC100 **Eğitilmiş ağırlıklarla başlangıç**	5-yönlü	
	1-örnek	5-örnek
Adım genişliği 0,1	54,54 ± 0,33	68,64 ± 0,26

Tiny ImageNet veri kümesiyle eğitilmiş modelin ağırlıklarının başlangıç durumu olarak kullanılması orijinal çalışma sonuçlarına yakın değerler elde edilmesini sağlamıştır ve bu sonuçlar Tablo 3.9'da paylaşılmıştır. Orijinal çalışmada başlangıç durumu olarak büyük ImageNet veri kümesi üzerinden elde edilmiş olan ağırlıkların kullanılması sonucunda 5-yönlü 1-örnek için 58,9 ± 1,9 doğruluk sonucu elde edilirken 5-yönlü 5-örnek için ise 71,5 ± 1,0 doğruluk sonucu elde edilmiştir.

Aradaki yaklaşık %3'lük farkın bu sebepten ötürü kabul edilebilir bir fark olduğu gözükmektedir.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine eğitilmiş ağırlıklar ile modelin başlatılması doğruluk oranlarında belirgin artışlara yol açmıştır. 5-yönlü 1-örnek için %12'lik bir artış gösterirken 5-yönlü 5-örnek için ise yaklaşık %9'luk bir artış göstermiştir.

Yapılan deneyler sonucunda, diğer veri kümelerinde olduğu gibi FC100 veri kümesi üzerinde de MAML algoritması için adım genişliğinin 0,2 veya 0,4 değerleri yerine 0,1 olarak seçilmesinin bir deney hariç bütün deney varyasyonlarında en iyi sonuçları verdiği gösterilmiştir. 5-yönlü deneylerine bakarsak eğer, adım genişliğinin 0,1 seçilmesi diğer adım genişliği seçimlerine göre sonuçlar üzerinde %5 ila %10 arasında bir iyileştirme göstermektedir. Verilen örnekte de belirtildiği üzere MiniImageNet veri kümesi üzerinde alınan sonuçlar arasındaki fark, eğer 5-yönlü (1-örnek ve 5-örnek) deneyleri için bakacak olursak %4 ila %7 arasında değişiklik göstermektedir. Aynı durum Omniglot veri kümesinde %8 ila %27 arasında değişiklik göstermekteydi.

Eğitim için 1 adım tercih edildiğinde test sürecinde 5 adım yerine 10 adım kullanmanın diğer veri kümeleri üzerinde olduğu gibi CUB veri kümesi üzerinde alınan sonuçları da arttırdığı gözlemlenmektedir. Aynı durum eğitim için 5 adım kullanıldığı zaman da geçerlidir. Ama bu artış genellikle %1 seviyelerinde kalmıştır, daha fazla bir iyileştirme elde edilememiştir. Bu olumlu yansımanın yüzdesi ise tercih edilen adım genişliğine göre farklılıklar göstermekle birlikte daha büyük adım genişliğinin (0,2 veya 0,4) tercih edildiği deneylerde olumlu yansıma da yaklaşık %3'lere çıkarak daha fazla gözükmektedir.

5-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen görüntü sayısının arttırılması doğruluk oranını da yükseltmektedir. Adım genişliğinin 0,1 olarak alındığı ilgili satırlara bakarsak eğer aradaki farkın bütün tablolarda yaklaşık olarak %15 olduğunu gözlemleyebiliriz.

Tablo 3.10'da ise aynı parametreler kullanıldığında kurmuş olduğumuz deneylerin sonuçları ile (Bertinetto vd., 2018) makalesindeki MAML için aynı veri

kümesi üzerinde alınan deneylerin sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.10: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

FC100	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	58,9 ± 1,9	71,5 ± 1,0
Kurulan deney düzeneği	54,54 ± 0,33	68,64 ± 0,26

3.2.4. Caltech-UCSD Birds Veri Kümesi Üzerindeki Sonuçları

Caltech-UCSD Birds (CUB) veri kümesi için, rastgele ağırlıklarla MAML algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.11'de verilmiştir. Her bir deney için 3 farklı adım genişliği ve ayrıca 4 farklı adım sayısı yapılandırması test edilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.11: MAML algoritmasının CUB veri kümesi doğruluk oranları.

Adım sayıları	**Adım genişliği**	5-yönlü		10-yönlü	
		1-örnek	5- örnek	1- örnek	5- örnek
Eğitim: 1 Test: 5	0,1	**42,05 ± 0,30**	**57,20 ± 0,27**	26,23 ± 0,16	**42,65 ± 0,16**
	0,2	40,45 ± 0,30	54,13 ± 0,27	**26,53 ± 0,17**	40,95 ± 0,16
	0,4	37,79 ± 0,28	49,62 ± 0,27	25,21 ± 0,17	30,49 ± 0,14
Eğitim: 1 Test: 10	0,1	**42,13 ± 0,30**	**57,63 ± 0,27**	26,22 ± 0,17	**42,23 ± 0,17**
	0,2	40,65 ± 0,30	54,86 ± 0,28	**27,43 ± 0,18**	40,30 ± 0,16
	0,4	37,54 ± 0,28	50,37 ± 0,28	25,37 ± 0,17	33,19 ± 0,15
Eğitim: 5 Test: 5	0,1	**45,22 ± 0,31**	**60,06 ± 0,27**	28,93 ± 0,17	**45,88 ± 0,17**
	0,2	41,94 ± 0,30	58,71 ± 0,27	**30,26 ± 0,18**	44,35 ± 0,16
	0,4	37,83 ± 0,27	55,47 ± 0,28	28,80 ± 0,17	38,64 ± 0,17
Eğitim: 5 Test: 10	0,1	**45,64 ± 0,31**	**60,30 ± 0,27**	29,75 ± 0,18	**46,29 ± 0,17**
	0,2	42,85 ± 0,31	59,23 ± 0,27	**30,42 ± 0,18**	44,37 ± 0,17
	0,4	38,17 ± 0,28	55,43 ± 0,28	28,01 ± 0,17	40,54 ± 0,17

Rastgele ağırlıklarla başlatmanın önceden eğitilmiş modelin ağırlıklarını kullanarak başlamayla arasındaki farkı görebilmek için bazı deneyler eğitilmiş ağırlıklarla başlayacak şekilde ayarlanıp tekrar edilmiştir. Eğitilmiş ağırlıklar ise, aynı yapı kullanılarak klasik model eğitimi düzeneğini kurarak (meta-learning eğitim sürecinden farklı olarak), Tiny ImageNet veri kümesi üzerinden modelin eğitilmesi ile elde edilmiştir.

Tablo 3.12: Eğitilmiş ağırlıkla başlangıç için alınan doğruluk oranları.

CUB **Eğitilmiş ağırlıklarla başlangıç**	5-yönlü	
	1-örnek	5-örnek
Adım genişliği 0,1	52,46 ± 0,34	67,92 ± 0,27

Tiny ImageNet veri kümesiyle eğitilmiş modelin ağırlıklarının başlangıç durumu olarak kullanılması orijinal çalışma sonuçlarına yakın değerler elde edilmesini sağlamıştır ve bu sonuçlar Tablo 3.12'de paylaşılmıştır. Orijinal çalışmada başlangıç durumu olarak büyük ImageNet veri kümesi üzerinden elde edilmiş olan ağırlıkların kullanılması sonucunda 5-yönlü 1-örnek için 55,92 ± 0,95 doğruluk sonucu elde edilirken 5-yönlü 5-örnek için ise 72,09 ± 0,76 doğruluk sonucu elde edilmiştir. Aradaki %4'lük farkın bu sebepten ötürü kabul edilebilir bir fark olduğu gözükmektedir.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine eğitilmiş ağırlıklar ile modelin başlatılması doğruluk oranlarında belirgin artışlara yol açmıştır. Sonuçlarına bakıldığında, 5-yönlü deneyleri için %8'lik bir artış olduğu gözükmektedir.

Yapılan deneyler sonucunda, diğer veri kümelerinde olduğu gibi CUB veri kümesi üzerinde de MAML algoritması için adım genişliğinin 0,2 veya 0,4 değerleri yerine 0,1 olarak seçilmesinin bir deney hariç bütün deney varyasyonlarında en iyi sonuçları verdiği gösterilmiştir. CUB veri kümesi üzerinde alınan sonuçlara göz gezdirirsek eğer, 5-yönlü deneyleri için (1-örnek ve 5-örnek) %5 ila %8 arasında iyileşme göstermektedir. Aynı durum Omniglot veri kümesinde %8 ila %27 arasında değişiklik göstermekteydi.

Eğitim için 1 adım tercih edildiğinde test sürecinde 5 adım yerine 10 adım kullanmanın diğer veri kümeleri üzerinde olduğu gibi CUB veri kümesi üzerinde alınan sonuçları da arttırdığı gözlemlenmektedir. Aynı durum eğitim için 5 adım kullanıldığı zaman da geçerlidir. Ama bu artış genellikle %1 seviyelerinde kalmıştır, daha fazla bir iyileştirme elde edilememiştir. Bu olumlu yansımanın yüzdesi ise tercih edilen adım genişliğine göre farklılıklar göstermekle birlikte daha büyük adım genişliğinin (0,2 veya 0,4) tercih edildiği deneylerde olumlu yansıma da yaklaşık %3'lere çıkarak daha fazla gözükmektedir.

5-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen görüntü sayısının arttırılması doğruluk oranını da yükseltmektedir. Adım genişliğinin 0,1 olarak alındığı ilgili satırlara bakarsak eğer aradaki farkın bütün tablolarda yaklaşık olarak %15 olduğunu gözlemleyebiliriz.

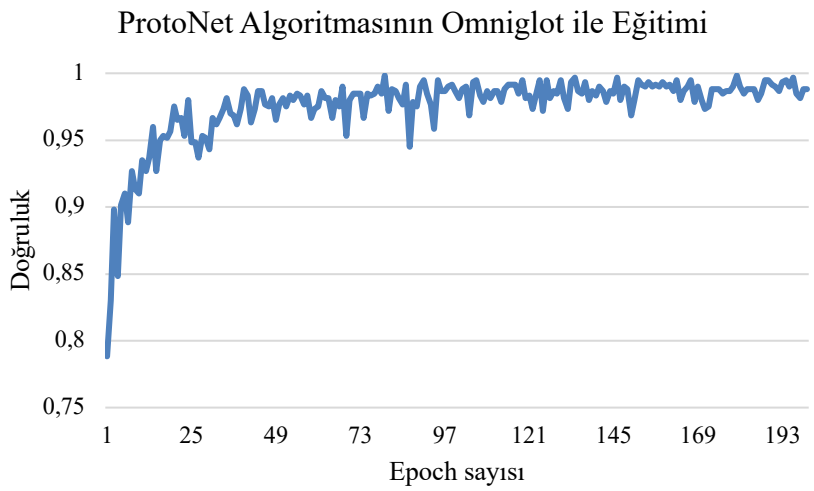
Tablo 3.13'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deneylerin sonuçları ile (Antoniou & Storkey, y.y.) makalesindeki MAML için aynı veri kümesi üzerinde alınan deneylerin sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.13: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

CUB	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	55,92 ± 0,95	72,09 ± 0,76
Kurulan deney düzeneği	52,46 ± 0,34	67,92 ± 0,27

3.3. PROTONET SONUÇLARI

MAML algoritmasında olduğu gibi, bu deneyler için de 200 epoch boyunca model eğitilmiştir. Ayrıca Şekil 3.4'de, 5-yönlü 5-örnek yapılandırmasının Omniglot veri kümesi üzerindeki eğitim sürecine ait doğruluk oranlarını içeren gerçek verilerden oluşan bir gidişat bulunmaktadır ve epoch sayısının yeterli olduğunu göstermektedir.



Şekil 3.4: ProtoNet Algoritmasının Omniglot veri kümesiyle eğitim sürecinin gidişatı

3.3.1. Omniglot Veri Kümesi Üzerindeki Sonuçları

Omniglot veri kümesi için, rastgele ağırlıklarla ProtoNet algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.14'de verilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.14: ProtoNet algoritmasının Omniglot veri kümesi doğruluk oranları.

Omniglot	5-yönlü		10-yönlü	
	1-örnek	5-örnek	1-örnek	5-örnek
Rastgele başlangıç	91,94 ± 0,19	97,99 ± 0,06	90,72 ± 0,14	97,74 ± 0,05

5-yönlü ve 10-yönlü deneylerin sonuçlarına bakarsak eğer, eğitim için modele gösterilen örnek sayısının (shot) arttırılması doğruluk oranını yükseltmektedir. Eğitim için tek bir görüntü örneği verildiğinde test sonuçları her iki deney (5-yönlü ve 10-yönlü) için de yaklaşık olarak %91 olarak gözlemlenmiştir. Eğitim için verilen örnek görüntü sayısı 1'den 5'e çıkartıldığında (5-örnek) ise test sonuçları her iki deney (5-yönlü ve 10-yönlü) için de yaklaşık olarak %98 olarak gözlemlenmiştir.

Aradaki %7'lik artışın tek sebebi görüldüğü üzere eğitim sürecinde verilen örnek görüntü sayısının 1'den 5'e çıkarılmasıdır. Bu durum hem 5-yönlü hem de 10-yönlü olarak yapılan deneylerde açıkça gözükmektedir. MAML algoritmasında da aynı durum gözlemlenmişti ve eğitim sürecinde eğer daha fazla örnek görüntü verilirse testin doğruluğunun artacağı yapılan deneylerde gösterilmişti.

Tablo 3.15'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deney sonuçları ile orijinal makalesindeki (Snell vd., 2017) deney sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.15: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

Omniglot	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	98,5 ± 0.2	99,5 ± 0.1
Kurulan deney düzeneği	94,66 ± 0,15	97,99 ± 0,06

Tablo 3.15'de paylaşılan 5-yönlü 1-örnek deney sonucu, 200 epoch sayısının tek örnek verilen yapılandırma için yetersiz kalmasından ötürü 450 epoch çalıştırılarak elde edilen sonucu göstermektedir. 450 epoch sonucunda elde edilen model aynı test düzeneği ile test edildiğinde ise %95 güven aralığında 94,66 ± 0,15 doğruluk sonucunu vermiştir. Bu sonuçta olması gereken sonuçtan yaklaşık olarak %4 daha azdır ve bu fark kabul edilebilir bir fark olmakla birlikte ProtoNet algoritmasının kurduğumuz PyTorch deney düzeneği üzerinde doğru bir şekilde çalıştırılabildiğini göstermektedir.

Yapılan bu deneylere ek olarak, sadece 5-yönlü deneyler için geçerli olmakla birlikte, modelin temsil (embedding) boyutu ve saklı katman (hidden) boyutu parametreleri ayrı ayrı olarak 64 yerine 128 değeri ile çalıştırılmış ve Tablo 3.16'da paylaşılan sonuçlar elde edilmiştir. Temsil boyutu TB olarak, saklı katman boyutu ise SK olarak kısaltılmıştır. Sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda hem 1-örnek hem de 5-örnek deneylerinde yaklaşık olarak %1'lik bir artış gözlemlenmiştir. Aynı artış oranı sadece saklı katman boyutunun 64 değerinden 128 değerine çıkartılması sonucunda da elde edilmiştir.

Tablo 3.16: ProtoNet algoritmasının Omniglot veri kümesi üzerindeki ek parametre sonuçları.

Başlangıç	Örnek sayısı	5-yönlü		
		TB: 64 – SK: 64	TB: 128 – SK: 64	TB: 64 – SK: 128
Rastgele	1-örnek	91,94 ± 0,19	92,23 ± 0,18	93,02 ± 0,17
	5-örnek	97,99 ± 0,06	98,18 ± 0,06	98,36 ± 0,05

3.3.2. MiniImageNet Veri Kümesi Üzerindeki Sonuçları

MiniImageNet veri kümesi için hem rastgele ağırlıklarla hem de Tiny ImageNet veri kümesi kullanılarak önceden eğitilmiş olan modelin ağırlıkları ile ProtoNet algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.17'de verilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.17: ProtoNet algoritmasının MiniImageNet veri kümesi doğruluk oranları.

MiniImageNet	5-yönlü		10-yönlü	
	1-örnek	5-örnek	1-örnek	5-örnek
Rastgele	26,83 ± 0,17	45,06 ± 0,22	16,98 ± 0,11	31,14 ± 0,13
Eğitilmiş	38,25 ± 0,25	50,62 ± 0,23	20,48 ± 0,12	35,88 ± 0,14

5-yönlü ve 10-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen örnek sayısının (shot) arttırılması doğruluk oranını yükseltmektedir. Sadece örnek sayısının bahsedilen şekilde arttırılması bütün deney sonuçlarını yaklaşık olarak 2 katına kadar çıkarmıştır. Bu durum hem 5-yönlü hem de 10-yönlü olarak yapılan deneylerde açıkça gözükmektedir. MAML algoritmasında da aynı durum gözlemlenmişti ve eğitim sürecinde eğer daha fazla örnek görüntü verilirse testin doğruluğunun artacağı yapılan deneylerde gösterilmişti.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine Tiny ImageNet veri kümesiyle eğitilmiş ağırlıklar ile modelin başlatılması doğruluk oranlarında artışlara yol açmıştır. 5-yönlü 1-örnek için %12'lik bir artış gösterirken 5-yönlü 5-örnek için

ise yaklaşık %5'lik bir artış göstermiştir. Bu artış düzeni MAML algoritmasının yine aynı veri kümesi üzerindeki sonuçları için de benzer şekildeydi.

Tiny ImageNet veri kümesiyle eğitilmiş modelin ağırlıklarının başlangıç durumu olarak kullanılması orijinal çalışma sonuçlarına yakın değerler elde edilmesini sağlamıştır. Orijinal çalışmada başlangıç durumu olarak büyük ImageNet veri kümesi üzerinden elde edilmiş olan ağırlıkların kullanılması sonucunda 5-yönlü 1-örnek için 42,90 ± 0,6 doğruluk sonucu elde edilmiştir. Elde ettiğimiz sonuçla orijinal makalede paylaşılan sonucun arasındaki yaklaşık %4'lük farkın bu sebepten ötürü kabul edilebilir bir fark olduğu gözükmektedir.

Orijinal çalışmada ayrıca 5-yönlü 5-örnek için test edilecek model eğitim aşamasında normal düzenden farklı olarak 20-yönlü 5-örnek olarak eğitilmektedir. Kurmuş olduğumuz deneyler için bu düzeni değiştirmeyerek diğer deneyler için geçerli olan düzende eğitim ve test gerçekleştirildi. Bunun sonucunda ise elde edilen sonuç orijinal makalede verilmiş olan 65.4 ± 0.5 sonucundan yaklaşık olarak %15'lik bir kayıp göstermektedir. Bunun üzerine, epoch sayısı arttırılarak gidişat izlenmiş ve daha fazla epoch sayısı ile yakın değerlere ulaşabildiği tespit edilmiştir. 1000 epoch sonucunda elde edilen model aynı test düzeneği ile test edildiğinde ise %95 güven aralığında 60,1 ± 0,25 doğruluk sonucunu vermiştir. Bu sonuçta olması gereken sonuçtan yaklaşık olarak %5 daha azdır ve bu fark kabul edilebilir bir fark olmakla birlikte ProtoNet algoritmasının kurduğumuz PyTorch deney düzeneği üzerinde doğru bir şekilde çalıştırılabildiğini göstermektedir.

Tablo 3.18'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deney sonuçları ile orijinal makalesindeki deney sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.18: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

MiniImageNet	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	42,90 ± 0,6	65,4 ± 0,5
Kurulan deney düzeneği	38,25 ± 0,25	60,1 ± 0,25

Yapılan bu deneylere ek olarak, sadece 5-yönlü deneyler için geçerli olmakla birlikte, modelin temsil boyutu ve saklı katman boyutu parametreleri ayrı ayrı olarak 64 yerine 128 değeri ile çalıştırılmış ve Tablo 3.19'da paylaşılan sonuçlar elde edilmiştir. Temsil boyutu TB olarak, saklı katman boyutu ise SK olarak kısaltılmıştır. Rastgele başlangıç deneyleri için, sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda 1-örnek deneyinde yaklaşık olarak %2'lik bir artış gözlemlenirken 5-örnek deneyinde ise yaklaşık olarak %2'lik bir düşüş gözlemlenmiştir.

Tablo 3.19: ProtoNet algoritmasının MiniImageNet veri kümesi üzerindeki ek parametre sonuçları.

Başlangıç	**Örnek sayısı**	5-yönlü		
		TB: 64 – SK: 64	TB: 128 – SK: 64	TB: 64 – SK: 128
Rastgele	1-örnek	26,83 ± 0,17	28,07 ± 0,18	28,45 ± 0,19
	5-örnek	45,06 ± 0,22	44,95 ± 0,23	42,86 ± 0,24
Eğitilmiş	1-örnek	38,25 ± 0,25	32,54 ± 0,22	32,41 ± 0,23
	5-örnek	60,10 ± 0,25	53,31 ± 0,25	51,98 ± 0,24

3.3.3. Fewshot-CIFAR100 Veri Kümesi Üzerindeki Sonuçları

Fewshot-CIFAR100 (FC100 olarak bahsedilecek) veri kümesi için hem rastgele ağırlıklarla hem de Tiny ImageNet veri kümesi kullanılarak önceden eğitilmiş olan modelin ağırlıkları ile ProtoNet algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.20'de verilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.20: ProtoNet algoritmasının FC100 veri kümesi doğruluk oranları.

FC100	5-yönlü		10-yönlü	
	1-örnek	5-örnek	1-örnek	5-örnek
Rastgele	40,86 ± 0,28	60,62 ± 0,28	27,49 ± 0,16	48,24 ± 0,18
Eğitilmiş	50,07 ± 0,31	67,67 ± 0,25	37,05 ± 0,19	56,47 ± 0,17

5-yönlü ve 10-yönlü deney sonuçlarına bakarsak eğer, eğitim için modele gösterilen örnek sayısının (shot) arttırılması doğruluk oranını yükseltmektedir. Sadece örnek sayısının 1'den 5'e çıkartılması bütün deney sonuçlarında yaklaşık olarak %20'lik bir iyileştirme göstermiştir. Bu durum hem 5-yönlü hem de 10-yönlü olarak yapılan deneylerde açıkça gözükmektedir. MAML algoritmasında da aynı durum gözlemlenmişti ve eğitim sürecinde eğer daha fazla örnek görüntü verilirse testin doğruluğunun artacağı yapılan deneylerde gösterilmişti.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine Tiny ImageNet veri kümesiyle eğitilmiş ağırlıklar ile modelin başlatılması, beklenildiği üzere doğruluk oranlarında artışlara yol açmıştır. 5-yönlü 1-örnek için %10'luk bir artış gösterirken 5-yönlü 5-örnek için ise yaklaşık %7'lik bir artış göstermiştir. 10-yönlü 1-örnek için %10'luk bir artış gösterirken 10-yönlü 5-örnek için ise yaklaşık %8'lik bir artış göstermiştir.

Tiny ImageNet veri kümesiyle eğitilmiş modelin ağırlıklarının başlangıç durumu olarak kullanılması orijinal çalışma sonuçlarına yakın değerler elde edilmesini sağlamıştır. Orijinal çalışmada başlangıç durumu olarak büyük ImageNet veri kümesi üzerinden elde edilmiş olan ağırlıkların kullanılması sonucunda 5-yönlü 1-örnek için 55,5 ± 0,7 doğruluk sonucu elde edilirken 5-yönlü 5-örnek için 72.0 ± 0.6 sonucu elde edilmiştir. Elde ettiğimiz sonuçlarla orijinal makalede paylaşılan sonuçların arasındaki yaklaşık %5'lük farkın bu sebepten ötürü kabul edilebilir bir fark olduğu gözükmektedir.

Tablo 3.21'de ise aynı parametreler kullanıldığında kurmuş olduğumuz deney sonuçları ile (Bertinetto vd., 2018) makalesindeki ProtoNet için aynı veri kümesi üzerinde alınan deneylerin sonuçlarının karşılaştırılması verilmiştir. Bu tablodan deney düzeneğimizin doğru bir şekilde çalıştığını gözlemleyebiliriz.

Tablo 3.21: Orijinal sonuçlarla alınan sonuçların karşılaştırılması.

FC100	5-yönlü	
	1-örnek	5-örnek
Orijinal makale	55,5 ± 0,7	72.0 ± 0.6
Kurulan deney düzeneği	50,07 ± 0,31	67,67 ± 0,25

Yapılan bu deneylere ek olarak, sadece 5-yönlü deneyler için geçerli olmakla birlikte, modelin temsil boyutu ve saklı katman boyutu parametreleri ayrı ayrı olarak 64 yerine 128 değeri ile çalıştırılmış ve Tablo 3.22'de paylaşılan sonuçlar elde edilmiştir. Temsil boyutu TB olarak, saklı katman boyutu ise SK olarak kısaltılmıştır. Rastgele başlangıç deneyleri için, sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda hem 1-örnek hem de 5-örnek deneylerinde yaklaşık olarak %1'lik bir düşüş gözlemlenmiştir. Sadece saklı katman boyutunun 64 değerinden 128 değerine çıkartılması sonucunda 1-örnek deneyinde yaklaşık olarak %2'lik bir artış gözlemlenirken 5-örnek deneyinde ise aynı sonuca ulaşıldığı gözlemlenmiştir. Eğitilmiş ağırlıklarla başlangıç deneylerinde ise daha farklı sonuçlar ortaya çıkmıştır. Eğitilmiş ağırlıklarla başlangıç deneyleri için, sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda 1-örnek deneyinde yaklaşık olarak %6'lık bir düşüş gözlemlenirken 5-örnek deneyinde ise yaklaşık olarak %4'lük bir düşüş gözlemlenmiştir.

Tablo 3.22: ProtoNet algoritmasının FC100 veri kümesi üzerindeki ek parametre sonuçları.

Başlangıç	**Örnek sayısı**	5-yönlü		
		TB: 64 – SK: 64	TB: 128 – SK: 64	TB: 64 – SK: 128
Rastgele	1-örnek	40,86 ± 0,28	40,27 ± 0,28	42,18 ± 0,28
	5-örnek	60,62 ± 0,28	59,64 ± 0,27	60,72 ± 0,28
Eğitilmiş	1-örnek	50,07 ± 0,31	49,65 ± 0,30	44,41 ± 0,29
	5-örnek	67,67 ± 0,25	70,09 ± 0,25	63,18 ± 0,26

3.3.4. Caltech-UCSD Birds Veri Kümesi Üzerindeki Sonuçları

Caltech-UCSD Birds (CUB olarak bahsedilecek) veri kümesi için hem rastgele ağırlıklarla hem de Tiny ImageNet veri kümesi kullanılarak önceden eğitilmiş olan modelin ağırlıkları ile ProtoNet algoritması başlatıldığında elde edilen doğruluk oranları Tablo 3.23'de verilmiştir. Bu verilen sonuçlar, eğitim veri kümesi üzerinde 200 epoch boyunca eğitilmiş olan modellerin sonrasında 600 epoch boyunca test veri kümesi üzerinde elde etmiş olduğu doğruluk oranlarının %95 güven aralığı içerisinde değerlendirilmesi ile elde edilmiştir.

Tablo 3.23: ProtoNet algoritmasının CUB veri kümesi doğruluk oranları.

CUB	5-yönlü		10-yönlü	
	1-örnek	5-örnek	1-örnek	5-örnek
Rastgele	34,07 ± 0,23	57,83 ± 0,26	21,93 ± 0,13	34,07 ± 0,23
Eğitilmiş	38,10 ± 0,27	61,56 ± 0,25	26,07 ± 0,16	47,48 ± 0,17

5-yönlü ve 10-yönlü sonuçlarına bakarsak eğer, eğitim için modele gösterilen örnek sayısının (shot) arttırılması doğruluk oranını yükseltmektedir. Sadece örnek sayısının 1'den 5'e çıkartılması 5-yönlü deney sonuçlarında yaklaşık olarak %23'lük bir iyileştirme gösterirken 10- yönlü deney sonuçlarında yaklaşık olarak %13'lük bir iyileştirme göstermiştir. Bu durum hem 5 sınıf hem de 10 sınıf olarak yapılan deneylerde açıkça gözükmektedir. MAML algoritmasında da aynı durum gözlemlenmişti ve eğitim sürecinde eğer daha fazla örnek görüntü verilirse testin doğruluğunun artacağı yapılan deneylerde gösterilmişti.

Rastgele başlangıç kullanarak sıfırdan başlatmak yerine Tiny ImageNet veri kümesiyle eğitilmiş ağırlıklar ile modelin başlatılması, doğruluk oranlarında artışlara yol açmıştır. 5-yönlü 1-örnek ve 5-yönlü 5-örnek deneyleri için %4'lük bir artış gösterirken, 10-yönlü deneylerinde bu artış daha farklı olmuştur. 10-yönlü 1-örnek için %5'lik bir artış gösterirken 10-yönlü 5-örnek için ise yaklaşık %13'lük bir artış göstermiştir.

Orijinal çalışmada bu veri kümesi üzerinde sadece örneksiz öğretim deneyleri yapılmış olup, 5-yönlü ve 10-yönlü deneyler 1-örnek ve 5-örnek yapılandırmaları için çalıştırılmamıştır. Sadece 50-yönlü 0-örnek deneyi sonuçları paylaşılmıştır.

Yapılan bu deneylere ek olarak, sadece 5-yönlü deneyler için geçerli olmakla birlikte, modelin temsil boyutu ve saklı katman boyutu parametreleri ayrı ayrı olarak 64 yerine 128 değeri ile çalıştırılmış ve Tablo 3.24'de paylaşılan sonuçlar elde edilmiştir. Temsil boyutu TB olarak, saklı katman boyutu ise SK olarak kısaltılmıştır. Rastgele başlangıç deneyleri için, sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda 1-örnek deneyinde yaklaşık olarak aynı sonuca ulaşıldığı gözlemlenirken 5-örnek deneyinde ise yaklaşık olarak %3'lük bir düşüş gözlemlenmiştir. Sadece saklı katman boyutunun 64 değerinden 128 değerine

çıkartılması sonucunda 1-örnek deneyinde yaklaşık olarak %3'lük bir düşüş gözlemlenirken 5-örnek deneyinde ise yaklaşık olarak %6'lık bir düşüş gözlemlenmiştir. Eğitilmiş ağırlıklarla başlangıç deneylerinde ise daha farklı sonuçlar ortaya çıkmıştır. Eğitilmiş ağırlıklarla başlangıç deneyleri için, sadece temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda hem 1-örnek hem de 5-örnek deneylerinde yaklaşık olarak aynı sonuçlara ulaşıldığı gözlemlenmiştir. Sadece saklı katman boyutunun 64 değerinden 128 değerine çıkartılması sonucunda da yine aynı şekilde hem 1-örnek hem de 5-örnek deneylerinde yaklaşık olarak aynı sonuçlara ulaşıldığı gözlemlenmiştir.

Tablo 3.24: ProtoNet algoritmasının CUB veri kümesi üzerindeki ek parametre sonuçları.

Başlangıç	**Örnek sayısı**	5-yönlü		
		TB: 64 – SK: 64	TB: 128 – SK: 64	TB: 64 – SK: 128
Rastgele	1-örnek	34,07 ± 0,23	33,74 ± 0,23	31,17 ± 0,20
	5-örnek	57,83 ± 0,26	54,90 ± 0,25	51,24 ± 0,26
Eğitilmiş	1-örnek	38,10 ± 0,27	38,50 ± 0,26	38,21 ± 0,26
	5-örnek	61,56 ± 0,25	60,72 ± 0,26	61,23 ± 0,26

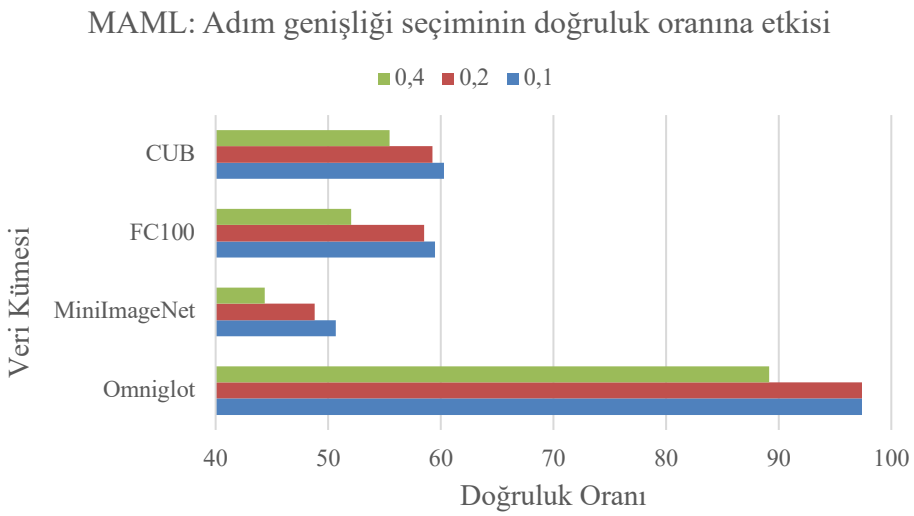
4. DENEY SONUÇLARININ DEĞERLENDİRİLMESİ

Yapılan deneyler sonucunda, bütün veri kümeleri için genellenebilir aşağıdaki sonuçlara ulaşılmıştır. Sonuçlar yüzdesel olarak farklılıklar gösterse de gidişat ve düzen olarak tamamen aynıdır. İlk değerlendirmeler, kullanılan veri kümesinden bağımsız olarak MAML algoritmasının deney sonuçları üzerinde yapılacaktır. Sonrasında ise ProtoNet algoritması sonuçlarının değerlendirmeleri paylaşılacaktır. Son olarak ise deneyler sırasında karşılaşılan zorluklardan bahsedilecektir.

MAML algoritması için adım genişliğinin 0,2 veya 0,4 değerleri yerine 0,1 olarak seçilmesinin bütün deney varyasyonlarında en iyi sonuçları verdiği deneysel sonuçlar bölümünde gösterilmiştir. Deneyler bölümündeki 5-yönlü 5-örnek deneylerinin eğitimde 5 testte 10 adım kullanılan yapılandırmalarının doğruluk oranları, bahsedilen durumun veri kümesinden bağımsız bir şekilde gerçekleştiğinin gözükebilmesi için Tablo 4.1'de toplu halde verilmiştir. Verilen bu özet tablo, diğer yapılandırmalar için de geçerli olmakla birlikte sadece yüzdesel artış oranları farklılık göstermektedir. Özellikle adım genişliği 0,4 olduğunda aradaki fark açıkça gözükmektedir ve daha kötü sonuçlar elde edilmektedir. Örnek olarak SGD'nin eğitimde 1 adım testte ise 5 adım çalıştırıldığı, yani öğrenmenin daha da zorlaştırıldığı durumda, 5-yönlü 1-örnek için yapılan deney sonuçlarını içeren sütuna bakarsak eğer, adım genişliği 0,4 iken elde edilen doğruluk 60,58 olurken adım genişliği 0,1 olarak alındığında doğruluk 87,77 gibi büyük bir yükseliş göstermiştir. Aralarında yaklaşık %27'lik bir fark bulunmaktadır ve bu farkın tek sebebi adım genişliğinin 0,4 yerine 0,1 olarak alınmış olmasıdır. Bu düşüşün sebebini MAML algoritmasında kullanılan SGD'nin daha büyük adımlarla ilerlendiğinde daha fazla sapma göstermeye meyilli olmasına bağlayabiliriz.

Tablo 4.1: MAML algoritması için adım genişliği seçiminin doğruluk oranına etkisi.

Adım genişliği	5-yönlü 5-örnek			
	Omniglot	MiniImageNet	FC100	CUB
0,1	**97,42 ± 0,07**	**50,69 ± 0,25**	**59,49 ± 0,29**	**60,30 ± 0,27**
0,2	97,39 ± 0,03	48,78 ± 0,26	58,49 ± 0,29	59,23 ± 0,27
0,4	89,15 ± 0,09	44,38 ± 0,27	52,06 ± 0,30	55,43 ± 0,28



Şekil 4.1: MAML için adım genişliği seçiminin doğruluk oranına etkisi

Tablo 4.1'de verilen özet tablo ayrıca diyagram halinde Şekil 4.1'de verilmiştir. Bu diyagram üzerinden genel gidişat daha net bir şekilde okunabilmektedir ve adım genişliği seçiminin etkisini belirgin bir şekilde ortaya koymaktadır. Aynı şekilde, MAML algoritması için en iyi sonucun adım genişliğinin 0,1 olarak alındığı deneyler sonucunda elde edildiği gösterilmiştir.

Eğitim için 1 adım tercih edildiğinde test sürecinde 5 adım yerine 10 adım kullanmanın sonuçları arttırdığı deneysel sonuçlar bölümünde gösterilmiştir. Aynı durum eğitim için 5 adım kullanıldığı zaman da geçerlidir. Sonuç olarak; eğitim için kullanılan adım sayısını sabitlediğimizde, test için kullanılacak olan adım genişliğinin arttırılmasının sonuçlara olumlu olarak yansıdığı gözlemlenmiştir. Bu olumlu yansımanın yüzdesi ise tercih edilen adım genişliğine göre farklılıklar göstermekle birlikte daha büyük adım genişliğinin (0,2 veya 0,4) tercih edildiği deneylerde olumlu yansıma da daha fazla gözükmektedir. Çünkü adım genişliği uygun değerinden (MAML için 0,1) uzaklaştıkça atılan adımların hedeften uzaklaşma olasılığı da artmaktadır, bu uzaklaşmayı atılacak adım sayısını arttırarak azaltabilmek mümkündür deney sonuçlarında da görüldüğü gibi. Ama eğer adım genişliği zaten uygun olarak seçilmişse (0,1 gibi) adım sayısının arttırılması sonuçlarda en fazla %1'lik bir iyileştirme göstermektedir. Tablo 4.2'de, bu iyileşmeye örnek olarak 5-yönlü 5-örnek deneylerinin, adım genişliği 0,4 olarak

alındığındaki sonuçlarının farklı veri kümelerinde almış olduğu doğruluk oranları paylaşılmıştır. Eğitim sürecinde 1 adım kullanılırken test sürecindeki adım sayısının 5 adımdan 10 adıma çıkmasının sonuçları ilk iki satırda gösterilmeye çalışılırken son iki satırda ise eğitim sürecinde 5 adım kullanılırken test sürecindeki adım sayısının 5 adımdan 10 adıma çıkmasının sonuçları gösterilmiştir.

Tablo 4.2: MAML algoritması için testteki adım sayısının doğruluk oranına etkisi.

Eğitim Adım Sayısı	**Test Adım Sayısı**	5-yönlü 5-örnek			
		Omniglot	MiniImageNet	FC100	CUB
1 adım	5 adım	81,45 ± 0,14	42,52 ± 0,24	49,38 ± 0,28	49,62 ± 0,27
	10 adım	**91,11 ± 0,10**	**42,63 ± 0,25**	**49,45 ± 0,28**	**50,37 ± 0,28**
5 adım	5 adım test	78,52 ± 0,14	46,51 ± 0,26	**52,40 ± 0,28**	**55,47 ± 0,28**
	10 adım test	**89,15 ± 0,09**	**47,18 ± 0,27**	52,06 ± 0,30	55,43 ± 0,28

Eğitim için modele gösterilen görüntü sayısının arttırılması bekleneceği gibi doğruluk oranını yükseltmektedir. Deneysel sonuçlar bölümünde bulunan 5-yönlü sonuçlarında adım genişliğinin 0,1 olarak alındığı ilgili satırlara bakarsak eğer aradaki farkın bütün tablolarda yaklaşık olarak %10 ila %15 arasında olduğunu gözlemleyebiliriz. Adım genişliğinin uygun değeri olan 0,1 değerinden farklı olarak seçildiği durumlarda ise bu artışın yüzdesinin daha fazla (%12 ila %18 arasında) olduğu gözlemlenmektedir. Sonuç olarak, beklenildiği üzere, eğitimde kullanılan örnek sayısının arttırılmasının aynı zamanda modelin doğruluğunu da arttıracağı düşünüldüğünden bu durumun test sonuçlarına olumlu olarak yansıdığını söyleyebiliriz.

Tablo 4.3'deki tabloda ise, bu iyileşmeye örnek olarak 5-yönlü deneylerinin adım genişliğinin 0,1 ve 0,4 olarak alındığı, eğitimdeki adım sayısının 5, testteki adım sayısının ise 10 olarak sabitlendiği sonuçlarının farklı veri kümelerinde almış olduğu doğruluk oranları paylaşılmıştır. İlk iki satırda, sabitlenen yapılandırma için (5 adım eğitim, 10 adım test) modele verilen örnek sayısının 1 örnekten 5 örneğe çıkmasının adım genişliği 0,1 olarak alındığındaki sonuçları gösterilmeye çalışılırken son iki satırda ise sabitlenen yapılandırma için (5 adım eğitim, 10 adım test) modele verilen örnek sayısının 1 örnekten 5 örneğe çıkmasının adım genişliği 0,4 olarak alındığındaki sonuçları gösterilmiştir.

Tablo 4.3: MAML algoritması için verilen örnek sayısının doğruluk oranına etkisi.

Adım genişliği	**Örnek sayısı**	5-yönlü			
		Omniglot	MiniImageNet	FC100	CUB
0,1	1-örnek	87,96 ± 0,22	34,22 ± 0,24	42,65 ± 0,31	45,64 ± 0,31
	5-örnek	**97,42 ± 0,07**	**50,69 ± 0,25**	**59,49 ± 0,29**	**60,30 ± 0,27**
0,4	1-örnek	77,21 ± 0,15	30,49 ± 0,22	34,20 ± 0,28	38,17 ± 0,28
	5-örnek	**89,15 ± 0,09**	**44,38 ± 0,27**	**52,06 ± 0,30**	**55,43 ± 0,28**

MAML algoritması için geçerli olan, örnek sayısının arttırılmasının sonucu olarak doğruluk oranının da artması durumu ProtoNet için de geçerli olmuştur. ProtoNet algoritmasının 5-yönlü deneyler için rastgele başlangıçla 200 epoch çalıştırılması sonucunda farklı veri kümelerinde almış olduğu doğruluk oranları toplu bir şekilde Tablo 4.4'de paylaşılmıştır. Bu sonuçlar ayrıca bölüm 4 içerisindeki ilgili başlıklar altında ayrı ayrı da bulunabilir. 5-yönlü deneylerde örnek sayısının arttırılmasının etkisi aynı şekilde 10-yönlü deneylerin sonuçlarında da gözükmektedir. Omniglot gibi kolay sayılabilecek bir veri kümesi dışındakilere baktığımızda, eğitim sürecinde 1 örnek yerine 5 örnek kullanmak doğruluk oranlarında yaklaşık olarak %20 artış sağlamıştır. Bunun sebebi olarak ise örnek sayısının arttırılmasının bu algoritmanın oluşturacağı ortalama prototipin doğruluğunu da arttırıyor olması düşüncesidir.

Tablo 4.4: ProtoNet algoritması için verilen örnek sayısının doğruluk oranına etkisi.

Örnek sayısı	5-yönlü			
	Omniglot	MiniImageNet	FC100	CUB
1-örnek	91,94 ± 0,19	26,83 ± 0,17	40,86 ± 0,28	34,07 ± 0,23
5-örnek	**97,99 ± 0,06**	**45,06 ± 0,22**	**60,62 ± 0,28**	**57,83 ± 0,26**

ProtoNet algoritmasının 5-yönlü 1-örnek deneyleri için, temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda almış olduğu doğruluk oranları toplu bir şekilde Tablo 4.5'de paylaşılmıştır. Omniglot veri kümesi hariç bütün deneyler ayrıca eğitilmiş başlangıç ağırlıkları ile de yapılarak sonuçları paylaşılmıştır. Sonuçlara baktığımızda, temsil boyutunun arttırılmış olmasının aynı veri kümesi içerisinde bile farklı sonuçlar göstermekte olduğunu söyleyebiliriz. Denenmiş olan iki farklı temsil boyutuna göre 5-yönlü 1-örnek deneyleri için genel

geçer bir kanı ortaya konulamamıştır. Bunun üzerine 5-yönlü 5-örnek deneylerinin de yapılması kararlaştırılarak, yine aynı şekilde, Omniglot veri kümesi hariç bütün deneylerin ayrıca eğitilmiş başlangıç ağırlıkları ile başlatıldığında elde ettikleri doğruluk oranları Tablo 4.6'da toplu bir şekilde paylaşılmıştır.

Tablo 4.5: ProtoNet algoritması için temsil boyutunun doğruluk oranına etkisi (5-yönlü 1-örnek).

Başlangıç	**Temsil boyutu**	5-yönlü 1-örnek			
		Omniglot	MiniImageNet	FC100	CUB
Rastgele	64	91,94 ± 0,19	26,83 ± 0,17	**40,86 ± 0,28**	**34,07 ± 0,23**
	128	**92,23 ± 0,18**	**28,07 ± 0,18**	40,27 ± 0,28	33,74 ± 0,23
Eğitilmiş	64	-	**38,25 ± 0,25**	**50,07 ± 0,31**	38,10 ± 0,27
	128	-	32,54 ± 0,22	49,65 ± 0,30	**38,50 ± 0,26**

ProtoNet algoritmasının 5-yönlü 5-örnek deneyleri için, temsil boyutunun 64 değerinden 128 değerine çıkartılması sonucunda almış olduğu doğruluk oranlarına baktığımızda, yine aynı şekilde, temsil boyutunun arttırılmış olmasının aynı veri kümesi içerisinde bile farklı sonuçlar göstermekte olduğunu söyleyebiliriz. Sonuç olarak, yapılan 5-yönlü deney sonuçlarına bakarak temsil boyutunun 64 değerinden 128 değerine çıkartılmasının elde ettiği doğruluk oranları ile ilgili genel geçer bir kanı ortaya konulamamıştır.

Tablo 4.6: ProtoNet algoritması için temsil boyutunun doğruluk oranına etkisi (5-yönlü 5-örnek).

Başlangıç	**Temsil boyutu**	5-yönlü 5-örnek			
		Omniglot	MiniImageNet	FC100	CUB
Rastgele	64	97,99 ± 0,06	**45,06 ± 0,22**	**60,62 ± 0,28**	**57,83 ± 0,26**
	128	**98,18 ± 0,06**	44,95 ± 0,23	59,64 ± 0,27	54,90 ± 0,25
Eğitilmiş	64	-	**60,10 ± 0,25**	67,67 ± 0,25	**61,56 ± 0,25**
	128	-	53,31 ± 0,25	**70,09 ± 0,25**	60,72 ± 0,26

ProtoNet algoritmasının 5-yönlü 1-örnek deneyleri için, saklı katman boyutunun 64 değerinden 128 değerine çıkartılması sonucunda almış olduğu doğruluk oranları toplu bir şekilde Tablo 4.7'de paylaşılmıştır. Omniglot veri kümesi hariç bütün deneyler ayrıca eğitilmiş başlangıç ağırlıkları ile de yapılarak sonuçları

paylaşılmıştır. Sonuçlara baktığımızda, saklı katman boyutunun arttırılmış olmasının aynı veri kümesi içerisinde bile farklı sonuçlar göstermekte olduğunu söyleyebiliriz. Denenmiş olan iki farklı saklı katman boyutuna göre 5-yönlü 1-örnek deneyleri için genel geçer bir kanı ortaya konulamamıştır. Bunun üzerine 5-yönlü 5-örnek deneylerinin de yapılması kararlaştırılarak, yine aynı şekilde, Omniglot veri kümesi hariç bütün deneylerin ayrıca eğitilmiş başlangıç ağırlıkları ile başlatıldığında elde ettikleri doğruluk oranları Tablo 4.8'de toplu bir şekilde paylaşılmıştır.

Tablo 4.7: ProtoNet algoritması için saklı katman boyutunun doğruluk oranına etkisi (5-yönlü 1-örnek).

Başlangıç	**Saklı katman**	5-yönlü 1-örnek			
		Omniglot	MiniImageNet	FC100	CUB
Rastgele	64	91,94 ± 0,19	26,83 ± 0,17	40,86 ± 0,28	**34,07 ± 0,23**
	128	**93,02 ± 0,17**	**28,45 ± 0,19**	**42,18 ± 0,28**	31,17 ± 0,20
Eğitilmiş	64	-	**38,25 ± 0,25**	**50,07 ± 0,31**	38,10 ± 0,27
	128	-	32,41 ± 0,23	44,41 ± 0,29	**38,21 ± 0,26**

ProtoNet algoritmasının 5-yönlü 5-örnek deneyleri için, saklı katman boyutunun 64 değerinden 128 değerine çıkartılması sonucunda almış olduğu doğruluk oranlarına baktığımızda, yine aynı şekilde, saklı katman boyutunun arttırılmış olmasının aynı veri kümesi içerisinde bile farklı sonuçlar göstermekte olduğunu söyleyebiliriz. Sonuç olarak, yapılan 5-yönlü deney sonuçlarına bakarak saklı katman boyutunun 64 değerinden 128 değerine çıkartılmasının elde ettiği doğruluk oranları ile ilgili genel geçer bir kanı ortaya konulamamıştır.

Tablo 4.8: ProtoNet algoritması için saklı katman boyutunun doğruluk oranına etkisi (5-yönlü 5-örnek).

Başlangıç	**Temsil boyutu**	5-yönlü 5-örnek			
		Omniglot	MiniImageNet	FC100	CUB
Rastgele	64	97,99 ± 0,06	**45,06 ± 0,22**	60,62 ± 0,28	**57,83 ± 0,26**
	128	**98,36 ± 0,05**	42,86 ± 0,24	**60,72 ± 0,28**	51,24 ± 0,26
Eğitilmiş	64	-	**60,10 ± 0,25**	**67,67 ± 0,25**	**61,56 ± 0,25**
	128	-	51,98 ± 0,24	63,18 ± 0,26	61,23 ± 0,26

Tablo 4.9: MAML ve ProtoNet doğruluk oranlarının karşılaştırılması (5-yönlü 5-örnek).

Algoritma	5-yönlü 5-örnek			
	Omniglot	MiniImageNet	FC100	CUB
MAML	97,42 ± 0,07	50,69 ± 0,25	59,49 ± 0,29	60,30 ± 0,27
ProtoNet	**97,99 ± 0,06**	**60,10 ± 0,25**	**67,67 ± 0,25**	**61,56 ± 0,25**

Tablo 4.9'da toplu olarak verilen 5-yönlü 5-örnek deney sonuçlarına baktığımızda ProtoNet algoritmasının daha başarılı doğruluk oranları elde ettiğini söyleyebiliriz. Bunun sebebi olarak ise ProtoNet algoritmasının her bir sınıf için bir prototip oluşturup bu prototipleri yeni gelen sınıf verilerine göre güncellemesine bağlayabiliriz. MAML algoritmasının yaptığı gibi bütün sınıfları kolaylıkla ayırt edebilecek ortak bir başlangıç ağırlıkları bulmaktansa her bir sınıfa ait olabilecek bir prototip bulmaya çalışması daha basit ve daha iyi bir sonuç ortaya koymuştur. Ayrıca, her bir sınıf için prototip oluşturmaya çalıştığından ProtoNet algoritması örneksiz eğitim deneylerinde de sıklıkla kullanılmakta ve başarılı sonuçlar elde etmektedir (Snell vd., 2017).

Tablo 4.10: MAML ve ProtoNet doğruluk oranlarının karşılaştırılması (5-yönlü 1-örnek).

Algoritma	5-yönlü 1-örnek			
	Omniglot	MiniImageNet	FC100	CUB
MAML	87,96 ± 0,22	34,22 ± 0,24	42,65 ± 0,31	**45,64 ± 0,31**
ProtoNet	**91,94 ± 0,19**	**38,25 ± 0,25**	**50,07 ± 0,31**	38,10 ± 0,27

Tablo 4.10'da ise 5-yönlü 1-örnek deney sonuçları toplu bir şekilde verilmiştir. CUB veri kümesi hariç, diğer bütün veri kümeleri üzerinde alınan doğruluk oranlarına baktığımızda ProtoNet algoritmasının daha iyi olduğunu söyleyebiliriz. Aynı durum 5-yönlü 5-örnek deneylerinde de gözlemlenmişti. Kötü sonuç alınan CUB veri seti sadece kuş türlerini içerdiğinden ve bu kuş resimlerinin birbirine yakınlığı diğer veri kümelerine göre daha fazla olduğu için ProtoNet algoritmasının tek örnek üzerinden prototip oluşturma işlemi daha başarısız bir sonuç alınmasına sebep olmuştur. Tablo 4.9'a baktığımızda ise 1 örnek yerine 5 örnek

verildiğinde bu farkın kapandığını ve ProtoNet algoritmasının daha iyi bir sonuç elde ettiğini gözlemleyebiliriz.

Bu kısımdan itibaren, bölüm 4 altında bulunan deney sonuçlarını elde etmek için yapılan deneyler sırasında karşılaşılan zorlukların bazılarından bahsedilecektir. İlk zorluk meta-öğrenme algoritmaları için gereken veri düzeninin oluşturulmasıdır. Normal öğrenme süreçlerinden farklı olarak, bölüm 2.2'de de incelemesi yapılan görevlerin (task) oluşturulması gerekmektedir. Bu görevler veri kümesinin bütünü kullanılarak oluşturulmakta ve bu görevler oluşturulurken literatürde referans olarak alınmakta olan Vinyals ve arkadaşları tarafından az örnekle öğrenme için önerilmiş olan yapılandırma (Vinyals vd., 2016) kullanılmaktadır. Görev ayrımları bu yapılandırmaya uygun olarak yapılmaktadır ve bu süreç için birden fazla farklı çözüm sunulmuştur. Fakat sunulan bu çözümler algoritmaya bağımlı olmakla birlikte aynı veri kümesinin farklı bir meta-öğrenme algoritmasında test edilmesi için bu yapının tekrardan değiştirilmesi ve farklı olan algoritmaya uyarlanması gerekmektedir. Bu durum ise farklı algoritmaların aynı veri kümesi üzerinde adil bir şekilde karşılaştırılmasının yapılmasını ve karşılaştırma ortamının bütün algoritmalar için adil bir ortam olması gerekliliğinin sağlanmasını zorlaştırmaktadır. Tez sürecinde yapılan ilk deneylerde, bu veri oluşturma kısmının da adil olması için ayrıca çabalar sarf edilmiştir. Torchmeta kütüphanesinin kullanıma sunulması (Deleu vd., 2019) ve bu ortamı oluşturmayı hem kolaylaştırıyor hem de bütün algoritmalar için aynı düzeni sağlıyor olması sebebiyle veri oluşturma katmanı tamamen değiştirilerek bu kütüphaneye geçilmiştir.

İkinci bir zorluk olarak ise deneylerin çalıştırılması için gereken hesaplama gücünden bahsedilecektir. İlk başlarda deneyler, okulun sağlamış olduğu sunucu üzerinde çalıştırılmış ve sonuçlar alınmıştır. Fakat bu sunucunun hesaplama gücü düşük olduğundan ve bu sunucu üzerinde deneyleri çalıştırma süreleri uzamaya başladığından alternatif çözümler üretilmesi zorunlu hale gelmiştir. Alternatif çözüm olarak ise farklı seçeneklere bakılmış ve sonuç olarak ise hem ücretsiz hem de her yerden rahatlıkla erişilebilir bir bulut hesaplama çözümü olan Google Colab seçeneğinde karar kılınmıştır. Bu sistemde de 12 saatlik zaman sınırı gibi belirli

kısıtlar olmasına rağmen bu sistemin bizim durumumuz için en uygun sistem olduğuna karar verilmiştir. Sonrasında ise kurulmuş olan bütün deneyler bu bulut hesaplama sistemine aktarılarak çalışır hale getirilmiştir. Deneylerin hepsi tekrarlanarak bu sistem üzerinden bütün sonuçlar elde edilmiştir.

5. SONUÇ

Eğitim sırasında öğrenilmiş olan görüntülerden farklı ama benzer olduğu düşünülen görüntülerle test edilirken, genelleme performansı bir evrişimsel sinir ağı (CNN) için genellikle daha düşüktür ve bu durum literatürde yapılan çalışmalarla kanıtlanmıştır. Bu nedenle, meta-öğrenme yaklaşımları daha da bir önem taşımaktadır ve bu açığı kapatmaya yönelik güncel olarak bu yönde iyileştirme çalışmaları yapılmaktadır. Çünkü klasik eğitim süreçleri sonrasında ortaya çıkan en temel sorun, modelin daha önce hiç karşılaşılmamış olan görüntülere adaptasyon zorluğudur, yani bir bakıma genelleme performansı düşüktür. Buna ek olarak, bir modelin sürekli olarak farklı veri kümeleri için sıfırdan eğitiliyor olması ise klasik eğitim süreçlerinin bir diğer önemli sorunlarından biridir. Meta-öğrenme yaklaşımları ile aynı yığın içerisinde birden fazla farklı görev verilerek modelin yeni verilecek olan görevlere de aynı şekilde genelleme performansı arttırılmaya çalışılır. Böylece model yeni ve daha önce görülmemiş görüntülere daha çabuk ve doğru bir şekilde adapte olabilir.

MAML algoritmasının aldığı sonuçları en çok etkileyen parametre adım genişliği olarak gözlemlenmiştir. Adım genişliği olması gereken değerden uzaklaşarak arttırıldığında sonuçlar üzerinde olumsuz etkileri görülmüştür. Bir başka durum ise modele eğitim sırasında gösterilen örnek sayısı ile ilgilidir, modele eğitim sırasında gösterilen örnek sayısı arttırıldığında hem MAML hem de ProtoNet daha başarılı sonuçlar elde etmiştir. Bütün parametrelerin değişken bir şekilde alınıp deneylerin yapılması elimizde bulunan hesaplama gücü (donanım gibi) bakımından mümkün olmadığı için iyi olduğu literatürde gösterilmiş olan veya bizim elimizde bulunan hesaplama gücüne uygun olan değerler belirlenmiştir. Örnek vermek gerekirse hem MAML hem de ProtoNet için daha küçük bir yığın boyutu tercih edilmiştir ki hesaplama sırasında elimizdeki kaynakların sınırı aşılmasın.

ProtoNet algoritmasının yaptığımız deneyler sonucunda MAML algoritmasından daha başarılı doğruluk oranları elde ettiği gözlemlenmiştir. ProtoNet algoritmasının her bir sınıf için farklı bir prototip oluşturup bu prototipleri yeni gelen sınıf verilerine göre güncellediği için daha başarılı olduğu sonucuna varılmıştır.

MAML algoritması bütün sınıfları kolaylıkla ayırt edebilecek ortak bir başlangıç ağırlıkları bulmaya çalışırken ProtoNet algoritmasının her bir sınıfa ait olabilecek farklı bir prototip bulmaya çalışması ortaya çıkan bu iyi sonucun ana nedenidir. Bu özelliğinden dolayı ProtoNet algoritması örneksiz eğitim deneylerinde de sıklıkla kullanılmakta ve başarılı sonuçlar elde etmektedir.

Daha sonraki çalışmalarda, ayrıca, backbone olarak kullanılan modeller değiştirildiğinde sonuçlara nasıl etki ettiğinin incelenmesi düşünülmektedir. Kullanılan ağ modeli derinleştikçe veya parametre sayısı arttıkça bu durumun modelin elde ettiği doğruluk oranlarına nasıl yansıyacağının gösterilmesi ayrıca önemlidir. Bu tez süresince backbone değiştirilmeden modelin saklı katman ve temsil boyutunda değişiklikler yapılarak ek deneyler yapılmış ve alınan bu deney sonuçlarından genel geçer bir kural ortaya koyulamamıştır. Sonraki çalışmalarda ayrıca aynı model üzerinde yapılacak parametre değişiklik yelpazesi (saklı katman ve temsil boyutu gibi) arttırılarak bu değişikliklerin sonuca nasıl yansıdığının gözlemlenmesi planlanmaktadır.

Bu tez süresince, yapılan deneylerde meta-öğrenme algoritmalarının sağlık veri kümeleri üzerindeki sonuçları araştırılmamıştır. Fakat gelecekteki çalışmalarda kullanılmak üzere tez süresince ayrıca meta-öğrenme algoritmalarının uygulanabileceği az sayıda örnek içeren zorlu veri kümeleri bulunmuştur. Tez sonrasında, meta-öğrenme algoritmalarının bu gibi alanlarda daha fazla keşfedilmesi üzerine çalışmalar yapılması düşünülmektedir. Bu alanların geliştirmeye daha açık olduğu gözükmektedir ve literatürde yapılmış olan çalışmalara bakıldığında bu eksiklik gözükmektedir. Bu eksikliğin bir diğer sebebi ise sağlık veri kümelerinin az örnekle öğrenme algoritmaları için gerekli olan sınıf ayrımlarını içermemesidir, genelde hastalıklı ve sağlıklı olarak sınıflandırma yapılmış olmasıdır. Bulunan uygun veri kümeleri için ayrıca gerekli düzenlemeler yapılarak meta-öğrenme algoritmalarının denenebileceği bir yapıda açık kaynak olarak yayımlanması planlanmaktadır.

KAYNAKÇA

Antoniou, A., Edwards, H., & Storkey, A. (2018). How to train your MAML. *7th International Conference on Learning Representations, ICLR 2019*. http://arxiv.org/abs/1810.09502

Antoniou, A., & Storkey, A. (y.y.). *Learning to Learn via Self-Critique*.

Bertinetto, L., Henriques, J. F., Torr, P. H. S., & Vedaldi, A. (2018). *Meta-learning with differentiable closed-form solvers*. 1–15. http://arxiv.org/abs/1805.08136

Biederman, I. (1987). Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*. https://doi.org/10.1037/0033-295X.94.2.115

Brazdil, P. B., Soares, C., & da Costa, J. P. (2003). Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. *Machine Learning*. https://doi.org/10.1023/A:1021713901879

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., & Huang, J.-B. (2019). *A Closer Look at Few-shot Classification*. http://arxiv.org/abs/1904.04232

Deleu, T., Würfl, T., Samiei, M., Cohen, J. P., & Bengio, Y. (2019). *Torchmeta: A Meta-Learning library for PyTorch*. http://arxiv.org/abs/1909.06576

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. *31st International Conference on Machine Learning, ICML 2014*.

Fei-Fei, L., Fergus, R., & Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2006.79

Fergus, R., Perona, P., & Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2003.1211479

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *34th International Conference on Machine Learning, ICML 2017*, *3*, 1856–1868. http://arxiv.org/abs/1703.03400

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on*

Machine Learning, ICML 2015.

Kalousis, A., & Hilario, M. (2000). Model selection via meta-learning: A comparative study. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*. https://doi.org/10.1109/TAI.2000.889901

Kemp, C., Perfors, A., & Tenenbaum, J. B. (2007). Learning overhypotheses with hierarchical Bayesian models. *Developmental Science*. https://doi.org/10.1111/j.1467-7687.2007.00585.x

Kim, J., Kim, S., & Choi, S. (2017). Learning to warm-start Bayesian hyperparameter optimization. *arXiv preprint arXiv:1710.06219*.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. *... Science Department, University of Toronto, Tech. ...*. https://doi.org/10.1.1.222.9220

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Lake, B. M., Salakhutdinov, R., Gross, J., & Tenenbaum, J. B. (2011). One shot learning of simple visual concepts. *In {Proceedings of the 33rd Annual Conference of the Cognitive Science Society}*.

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2019). The Omniglot challenge: a 3-year progress report. Içinde *Current Opinion in Behavioral Sciences*. https://doi.org/10.1016/j.cobeha.2019.04.007

Le, Y., & Yang, X. (y.y.). *Tiny ImageNet Visual Recognition Challenge*.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. https://doi.org/10.1109/5.726791

Leite, R., & Brazdil, P. (2010). Active testing strategy to predict the best classification algorithm via sampling and metalearning. *Frontiers in Artificial Intelligence and Applications*. https://doi.org/10.3233/978-1-60750-606-5-309

Nichol, A., Achiam, J., & Schulman, J. (2018). *On First-Order Meta-Learning Algorithms*. 1–15. http://arxiv.org/abs/1803.02999

Oreshkin, B. N., Rodriguez, P., & Lacoste, A. (2018). TADAM: Task dependent adaptive metric for improved few-shot learning. *Advances in Neural Information Processing Systems*. http://arxiv.org/abs/1805.10123

Ravi, S., & Larochelle, H. (2017). Optimization as a Model for Few-Shot Learning. *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 1–11.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*. https://doi.org/10.1007/s11263-015-0816-y

Smith, L. B., Jones, S. S., Landau, B., Gershkoff-Stowe, L., & Samuelson, L. (2002).

Object name learning provides on-the-job training for attention. *Psychological Science*. https://doi.org/10.1111/1467-9280.00403

Snell, J., Swersky, K., & Zemel, R. S. (2017). Prototypical Networks for Few-shot Learning. *Advances in Neural Information Processing Systems*. http://arxiv.org/abs/1703.05175

Thrun, S., & Pratt, L. (1998). Learning to Learn: Introduction and Overview. Içinde *Learning to Learn*. https://doi.org/10.1007/978-1-4615-5529-2_1

Vanschoren, J. (2018). *Meta-Learning: A Survey*. 1–29. http://arxiv.org/abs/1810.03548

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, *Nips*, 3637–3645.

Wah, C., Branson, S., Welinder, P., Perona, P., & Belongie, S. (2011). *The Caltech-UCSD Birds-200-2011 Dataset*. http://www.birdfieldguide.com

Weber, M., Welling, M., & Perona, P. (2000). Unsupervised learning of models for recognition. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/3-540-45054-8_2

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*.