ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

Alişan AKTAY

PYTHON PROGRAMLAMA DİLİ KULLANILARAK HATAY METEOROLOJİ RADAR GÖRÜNTÜLERİNİN YAPAY SİNİR AĞI İLE TAHMİNİ

UZAKTAN ALGILAMA VE COĞRAFİ BİLGİ SİSTEMLERİ
ANABİLİM DALI

ADANA-2021

ÖZ

YÜKSEK LİSANS TEZİ

PYTHON PROGRAMLAMA DİLİ KULLANILARAK HATAY METEOROLOJİ RADAR GÖRÜNTÜLERİNİN YAPAY SİNİR AĞI İLE TAHMİNİ

Alişan AKTAY

ÇUKUROVA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
UZAKTAN ALGILAMA VE COĞRAFİ BİLGİ SİSTEMLERİ ANABİLİM DALI

Danışman : Doç. Dr. Nuri EMRAHOĞLU
Yıl: 2021, Sayfa: 108
Jüri : Doç. Dr. Nuri EMRAHOĞLU
: Doç. Dr. Bekir Yiğit YILDIZ
: Prof. Dr. Muhittin ŞAHAN

Küresel ısınma ile birlikte, dünyanın birçok yerinde olağan dışı hava olayları şiddetli yağışlar ve su taşkınları meydana gelmektedir. Küresel iklim krizi, kentselleşme ve hızlı nüfus artışı ile birlikte değerlendirildiğinde, tutarlı ve zamanında yapılıp erken uyarı niteliği taşıyan hava tahminlerinin hayati bir öneme sahip olduğu görülmektedir.

Bu tez çalışmasında, erken uyarılarda ve kısa süreli kuvvetli yağış tahminlerinde sıkça kullanılan ve bir uzaktan algılama cihazı olan meteoroloji radar verileri işlenmiş ve kullanılmıştır. Hatay meteoroloji radarından alınan PPI görüntüler, bir Yapay Sinir Ağı (YSA) algoritması olan ConvLSTM modeli ile Python programlama diliyle yazılan kodlarla eğitilmiştir. Geliştirilen model kullanılarak son 6 radar görüntüsü eğitilmiş ve henüz kayıt edilmemiş bir sonraki görüntünün tahmini yapılmıştır.

Veri seti için seçilen görüntü zaman aralığı 30 dakikadır. Model eğitim doğruluk oranı %80-84 aralığındadır. Model tahmin görüntüsü ile gerçekleşen görüntünün benzerlik oranları Kök Ortalama Kare Hatası (RMSE), Tepe Sinyali Gürültü Oranı (PSNR) ve Yapısal Benzerlik İndeksi (SSIM) ile ölçülmüştür. Uygulanan modelden elde edilen sonuçlara göre en iyi performansa sahip tahmin değerleri, RMSE 0.000013, PSNR 97.795 ve SSIM, 0.998 olarak hesaplanmıştır.

Kullanılan YSA modeliyle çeşitli zaman aralıkları ile sıralı radar görüntülerinin eğitilebileceği ve belirlenen zaman aralığı kadar sonra oluşacak görüntü tahminin yapılabileceği gösterilmiştir. Model doğruluk oranlarının artırılması için veri setinin artırılması ve çeşitlendirilmesi gerekmektedir.

Anahtar Kelimeler: Python, ConvLSTM, Uzaktan Algılama, Meteoroloji, Veri Bilimi

ABSTRACT

MSc THESIS

PREDICTION OF HATAY METEOROLOGY RADAR IMAGES WITH ARTIFICIAL NEURAL NETWORK USING PYTHON PROGRAMMING LANGUAGE

Alişan AKTAY

ÇUKUROVA UNIVERCITY
INSTITUTE OF NATURAL AND APPLIED SCIENCE
DEPARTMENT OF REMOTE SENSING AND GEOGRAPHICAL INFORMATION SYSTEM

Supervisor : Assoc. Prof. Dr. Nuri EMRAHOĞLU
Year: 2021, Pages: 108
Jury : Assoc. Prof. Dr. Nuri EMRAHOĞLU
: Assoc. Prof. Dr. Bekir Yiğit YILDIZ
: Prof. Dr. Muhittin ŞAHAN

In association with global warming, unusual weather events, heavy rains and floods occur in many parts of the world. When the global climate crisis is evaluated together with urbanization and rapid population growth, it is seen that making consistent and timely weather forecasts, which have the quality of early warning, have vital importance.

In this thesis, meteorology radar data, which is a remote sensing device used frequently in early warnings and short-term strong precipitation forecasts, has been processed and used. PPI images taken from Hatay meteorological radar are trained with codes written in Python programming language with ConvLSTM model, an Artificial Neural Network (ANN) algorithm. With the developed model, the last 6 radar images were trained and the next image, which has not yet been recorded, was estimated.

The image time interval selected for the dataset is 30 minutes. Model training accuracy is in the range of 80-84%. The similarity ratios of the model prediction image and the actual image were measured by Root Mean Square Error (RMSE), Peak Signal to Noise Ratio (PSNR), and Structural Similarity Index (SSIM). According to the results obtained from the applied model, the predictive values with the best performance were calculated as RMSE 0.000013, PSNR 97.795 and SSIM 0.998.

With the ANN model used, it has been shown that sequential radar images can be trained with various time intervals and the image prediction that will occur after the specified time interval can be made. In order to increase model accuracy, the data set should be increased and diversified

Key Words: Python, ConvLSTM, Remote Sensing, Meteorology, Data Science

GENİŞLETİLMİŞ ÖZET

Meteoroloji; dünyayı çevreleyen atmosferi anlama ve tahmin etme bilimidir. Meteoroloji bilimi, sosyal yaşamda, endüstriyel alanlarında, ulaşımda, tarım sektörü ve benzeri birçok alanda önemli bir etkiye sahiptir. Küresel ısınma; kentselleşme ve hızlı nüfus artışı ile birlikte değerlendirildiğinde, tutarlı bir hava tahmininin hayati bir öneme sahip olduğu görülmektedir.

Meteoroloji radarları sahip olduğu donanımları sayesinde ürettikleri elektromanyetik dalganın dönüşünü tekrar algılayarak uzaktan algılama işlevi gerçekleştirmektedir. Radarların aktif sensörler aracılığı ile gönderilen sinyalin dönüş süresi, gönderilen ve ışık hızında hareket eden sinyalin ne kadar süre sonra hedeften döndüğünü, ne kadar zayıflamış olduğunu, frekansında ve polarizasyonunda oluşan değişikliği algılayabilir. Elde edilen bu bilgiler doğrultusunda hedef nesnenin içeriği ve radar sahasına olan uzaklığı hesaplanabilir (Bakır, 2018).

Meteoroloji uygulamalarında tarama stili ile oluşturulan yansıma (reflektivite, dBZ) değerleri ile yağış veya yağışa dönüşebilecek su miktarı tahmini için Doppler radar ürünleri tercih edilmiştir. Bu durum kümülüform (küme) tip bulutlar yüksek oranda uyumlu bir yaklaşım olurken, stratiform (tabaka) tip bulutlar için yer gözlem verileri ile meydana gelen yağış miktarlarında ise hata oranları yüksek olmaktadır.

Dünya Meteoroloji Teşkilatı'nın (World Meteorological Organization, WMO) yağış ölçümü hususunda standartlara uygun olarak ölçüm yapan yağış ölçüm ağları bulunmaktadır. Radarlar doğrudan yağış ölçmezler ama ampirik bağıntılarla yağış bilgisini verirler. Bundan dolayı yağış ölçümü yapılmayan bölgelerde meteoroloji radarları kullanılarak yüksek çözünürlükte alansal yağış ölçümü bilgisi alınabilmektedir. Bu sebepten meteoroloji radarları geniş alanlardaki yağış dağılımını belirleyen en iyi uzaktan algılama ve gözlem aracı olarak kabul edilmektedir.

Meteoroloji radarları, yağış alanlarına dair gözlem imkânı sağlarken, zaman

içerisinde hareketinin takip edilmesi ve kısa vadede nerede bulunacağının tahmin edilebilmesi açısından kullanılan etkin uzaktan algılama cihazlarıdır. Radar aracılığıyla gönderilen elektromanyetik dalga sinyali, havada asılı duran su parçacıklarına çarparak yansır. Yansıyan bu eko sinyali gönderilen sinyale nazaran daha küçük boyutta radar tarafından tekrar algılanır. Radarlar sahip oldukları donanım ve yazılımlar sayesinde algıladıkları eko sinyalleri yükseltilerek reflektivite faktörü olarak adlandırılan "Z" değerinin hesaplanmasında kullanılmaktadır. Buna ek olarak meteoroloji radarları göndermiş oldukları elektromanyetik dalgaların frekans bant türüne göre adlandırılmaktadır. Meteoroloji radarları yağışın tipini, yağışın miktarını veya havada asılı damlacığın büyüklüğünü vb. parametrelerini, gönderdikleri güçlü elektromanyetik dalgaların hedeften geri dönen sinyalin göderilen sinyale göre farkına göre belirleyebilmektedirler.

Meteoroloji Genel Müdürlüğü 17 adet C-Bant, 2 adet ise X-Bant olmak üzere toplanda 19 adet meteoroloji radarını bünyesinde bulundurmaktadır. Bu çalışmada ise Meteoroloji Genel Müdürlüğü'nün Hatay İlinde konumlandırdığı meteoroloji radarı verileri kullanılmıştır.

Meteoroloji radarları en çok kısa vadeli hava tahminlerinde ve erken uyarılarda kullanılmaktadırlar. Kısa vadeli hava tahmini, meteorolojik parametrelerin tekdüze olmayan dağılımları ve kusurlu ya da hatalı ölçümler dolayısı ile en zorlu problemlerinden biridir. Sayısal Hava Tahmin Modellerine dayalı geleneksel tahminler modelin ilk oluşturulmasındaki uzman insan kaynağı ihtiyacı, gürültülü verilere karşı hassas oluşundan, zaman ve bilgisayar performansı açısından maliyetli olduğundan kısa vadeli tahminler için uygun bir yöntem değildir. Bu nedenlerden dolayı özellikle Yapay Sinir Ağları (YSA) ile hava durumu tahminine yönelik makine öğrenimi yaklaşımları, son zamanlarda aktif bir araştırma alanı olmuştur (Heye, 2017).

Kısa vadeli hava tahminleri için özellikle kuvvetli yağmur dolu oluşumlarında radar verisi çok işlevseldir. Radar görüntüleri aracılığı ile tahminci kuvvetli yağış alanının hareketini ve yönünü öngörebilir.

Bu tez çalışmasındaki amaç henüz kayıt edilmemiş radar görüntüsünü, geçmiş yani kayıt edilmiş radar görüntülerine bakılarak tahmin etmektir. Bu işlem içinde görüntü işleme konusunda başarılı sonuçlar elde eden Convolutional Neural Network (Evrişimli Sinir Ağları, CNN) ile birlikte, zaman serisi işlemlerinde başarılı sonuçlar elde edilmiş Long Short-Term Memory (Uzun - Kısa Süreli Bellek, LSTM) ağlarının birleşimi olan ConvLSTM algoritması kullanılmıştır.

TEŞEKKÜR

Yüksek lisans öğrenimim boyunca her türlü desteğini ve ilgisini benden esirgemeyen, danışman hocam sayın Doç. Dr. Nuri EMRAHOĞLU'na sonsuz teşekkürlerimi sunarım.

Bu süreçte her konuda olduğu gibi tez çalışması için de desteğini esirgemeyen, tecrübesinden ve bilgilerinden faydalandığım, her daim desteğini hissettiğim kıymetli hocam sayın Doç. Dr. Bekir Yiğit YILDIZ'a çok teşekkür ederim.

Tez aşamasında, özellikle radar donanım ve veri konusunda bilgi ve deneyimlerinden faydalandığım Meteoroloji Genel Müdürlüğü Uzaktan Algılama ve Şube Müdürlüğü çalışanlarına teşekkür ederim.

Ç.Ü. Fen Bilimleri Enstitüsü Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Bölümüne, maddi destek veren Ç.Ü. Bilimsel Araştırma Projeleri Birimi'ne teşekkürlerimi sunarım.

Tez yazım sürecinde; tecrübelerinden yaralandığım çalışma arkadaşım, devrem, dostum Ahmet HAMARAT'a, her sıkıştığımda aradığım, konuştuktan sonra motive olduğum, kapılarını çalmaktan çekinmediğim dostlarım Murat BAYAZIT ve Zuhal KURT'a desteklerinden dolayı teşekkür ederim. Radar üzerine çalışma yapan ve kaynaklarını, çalışmasını paylaşmaktan mutluluk duyan Ali Kemal BAKIR'a teşekkür ederim.

Yüksek lisansım boyunca her seferinde beni cesaretlendiren, hep yanımda hissettiğim, bazen beraber geçireceğimiz zamandan çaldığım, bazen sinirlendirdiğim hayat arkadaşım eşim Ayşe AKTAY'a teşekkür ediyorum. Bu süreçte doğan, büyüyen, bu aralar işine gelince konuşan, her bilgisayarı açtığımda gelip kurcalamak isteyen kapatan, evimizin neşesi, gürültücü canım oğlum Rüzgâr Ali AKTAY sana da teşekkürler. İyiki varsınız.

İÇİNDEKİLER **SAYFA**

ÇİZELGELER DİZİNİ **SAYFA**

ŞEKİLLER DİZİNİ **SAYFA**

SİMGELER VE KISALTMALAR

CAPPI : Constant Altitude Plan Position Indicator / Sabit Yükseklik PPI radar ürünü

CNN : Convolutional Neural Network / Evrişimli Sinir Ağı

ConvLSTM : Convolutional Long Short-Term Memory / Evrişimsel Uzun-Kısa Vadeli Bellek

CPU : Central Processing Unit / Merkezi İşlem Birimi

dBZ : Diferansiyel Reflektivite

EMS : Electromagnetic Spectrum / Elektromanyetik Spektrum

FC-LSTM : Fuly Connected LSTM / Tamamen bağlı LSTM hücresi

GDAL : Geospatial Data Abstraction Library Virtual Format

GeoTIFF : Geo-referenced Tagged Image File Format

GMT : Greenwich Mean Time / Greenwich Ortalama Zamanı

GPU : Graphics Processing Unit / Grafik İşlemci Birimi

Hdf : Hierarchical Data Format

IEEE : Institute of Electrical and Electronics Engineers

Iris : Interactive Radar Information System

ITU : Inertial Measurement Unit

LSTM : Long Short-Term Memory / Uzun-Kısa Vadeli Bellek

LSTM : Long short-term memory / Uzun-Kısa Vadeli Bellek

MAX : Maximum Display

MEVBİS : Meteorolojik Veri Bilgi Sunum ve Satış Sistemi

MGM : Meteoroloji Genel Müdürlüğü

MSE : Mean Squared Error / Ortalama Karesel Hata

NATO : North Atlantic Treaty Organization

NetCDF : Network Common Data Form

OMGİ : Otomatik Meteoroloji Gözlem İstasyonu

PPI : Plan Position Indicator

PSNR	: Tepe Sinyali Gürültü Oranı / Peak Signal to Noise Ratio
Radar	: Radio Detection and Ranging
Radar	: Radio Detection And Ranging
RAINN	: N saatlik RAIN radar ürünü (RAIN1 : 1 saatlik RAIN ürünü)
RHI	: Range Height Indicator
RMSE	: Root Mean Squared Error / Kök Ortalama Kare Hatası
RNN	: Yenilenen-Tekrarlanan Sinir Ağları / Recurrent Neural Network
SHT	: Sayısal Hava Tahmin
SSIM	: Yapısal Benzerlik Endeksi / Structural Similarity
UA	: Uzaktan Algılama
VIL	: Vertical Integrated Liquid / Dikey Entegre Sıvı
WMO	: World Meteorology Organization / Dünya Meteoroloji Teşkilatı
YSA	: Yapay Sinir Ağları
YTTS	: Yıldırım Tespit ve Takip Sistemi
Z	: Reflektivite
Z-R	: Reflektivite-Yağış Bağıntısı

1. GİRİŞ

1.1. Meteoroloji ve Tahmin

Meteoroloji, dünya atmosferinin yapısını, bileşimini, özelliklerini ve atmosferde meydana gelen hava olaylarını inceleyen bir bilim dalıdır (Ahrens, 2016).

Dünya üzerindeki yaşam; birçok etken ile birlikte büyük oranda varlığını atmosfere borçludur. Dolayısı ile bu varlığın devamı ya da zorluğu atmosfer içerisinde meydana gelen ya da gelebilecek olaylara bağlıdır. Meteoroloji bilimi meydana gelmiş olayları inceler ve meydana gelebilecek olayları yapısını fiziğini bildiği ya da ölçtüğü atmosfer parametrelerinden yola çıkarak tahmin eder.

Hava tahmini, atmosferin genel durumunun ölçülmesi, bu ölçümlerin geçmiş verilerle birlikte analiz edilmesi ile bir bölgede meydana gelebilecek hava olaylarının tahminidir. Hava tahmini insan sosyal yaşamında ve devamında çok önemli etkilere sahiptir. Son yıllarda, hızlı iklim değişikliği, olağandışı yağış nedeniyle dünyanın çeşitli yerlerinde ani seller gibi şiddetli meteorolojik olaylara da yol açmaktadır. İklim değişikliği ile birlikte tutarlı bir hava tahmininin önemi daha da artmıştır. Kimimiz için birkaç gün ya da saat sonra olacak olan yağışın anlamı; evden çıkarken şemsiye alıp almamak iken bir çiftçi içinse ürününü kaybetme olasılığı, bir tedarik zinciri için ürünün gecikmesi olabilir.

Hava tahmini kısa ve uzun vadeli olarak sınıflandırılabilir. Kısa vadeli hava tahmini, bir bölge için 0-6 saat aralığını kapsayan meteorolojik tahmindir (Doviak, 1994).

1.2. Meteorolojide Uzaktan Algılama

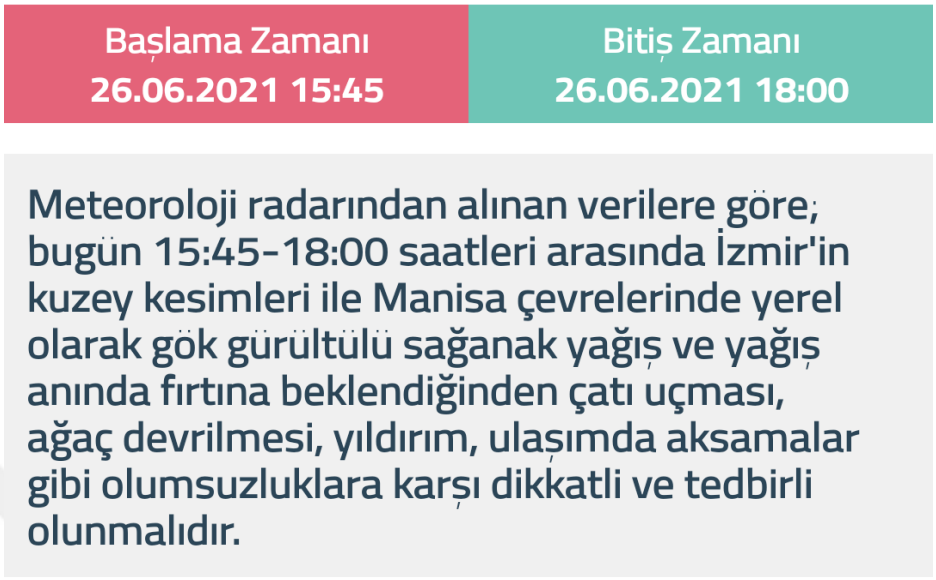
Hava tahmini; atmosfer parametrelerinin ölçülmesi ile başlar. Tahmin yapılabilmesi için atmosferin anlık görüntüsü alınması gerekir. Kısaca tahmin yapılacak yerin genel atmosfer şartları ve değişimleri bilinmelidir. Bu ölçüm işlemi, sıcaklık basınç, nem, rüzgâr, bulutluluk gibi atmosferi temsil eden birçok parametre

içerir. Bu ölçümler; insanlı ve insansız meteoroloji istasyonları tarafından ölçülürler.

Tutarlı bir tahmin için ölçümlerin yer yüzeyinde, okyanuslarda, denizlerde ve bütün atmosfer katmanlarında olması önemlidir. Bu kadar geniş bir alandan dikey ve yatay olarak ölçüm almak zor ve maliyeti büyüktür. Bu noktada uzaktan algılama devreye girmektedir. Çünkü uzaktan algılamanın temel amacı bir nesne hakkında bilgi alma işlemidir. Uzaktan algılama ile gidilemeyen yerlerden, insansız olarak daha az maliyetle ölçüm alınabilmektedir. Günümüzde hava tahminlerinin; tahmin başarısına yükselmesinin en büyük nedenlerinden biri uzaktan algılama cihazları ile insansız alanlar, çöller ve okyanuslardan elde edilen bilgilerin giriş değeri olarak Sayısal Hava Tahmin (SHT) modellerine eklenmesi ile sağlanmıştır.

Meteorolojide uzaktan algılama cihazları olarak meteoroloji radarları, meteoroloji uydular ve Yıldırım Tespit ve Takip Sistemi (YTTS) cihazları bulunmaktadır. Meteoroloji radarları birer aktif uzaktan algılama cihazı iken, meteoroloji uyduları pasif uzaktan algılama cihazlarıdır. Meteoroloji uyduları ile bulutluluk, aeresol, yüzey sıcaklığı gibi birçok meteorolojik parametre ölçülmektedir. Meteoroloji radarları ise bir alana yayılmış yağışı ve rüzgârın radyal hızını ölçmektedirler.

Meteoroloji Genel Müdürlüğü (MGM), meteoroloji radarlarını aktif olarak kullanmakta ve kısa süreli yağış tahminlerini radarı referans göstererek yapmaktadır. Şekil 1.1'de MGM tarafından meteoroloji radar verisine göre yapılmış bir uyarı görülmektedir.



Şekil 1.1. 26.06.2021 tarihinde İzmir ve Manisa için yapılan kuvvetli yağış uyarısı.[1]

1.3. Tahmin ve Yapay Zekâ

Hava tahminleri, atmosferin mevcut durumu hakkında nicel veriler toplayarak ve atmosferin nasıl gelişeceğini / değişeceğini tahmin etmek için atmosferik süreçlerin bilimsel işleyişi kullanarak yapılır (Hasan, 2019). Bu işlem günümüzde büyük oranda Sayısal Hava Tahmin (SHT) modelleri ile yapılmaktadır. Fakat atmosferin düzensiz ve karmaşık doğası, bir açık yara gibi her şeyden etkilenebilme durumu, başlangıç koşullarının ölçülmesinde yer alabilecek hatalı veri ile atmosferi tanımlayan denklemleri çözmek için muazzam hesaplama gücü ve zaman gerekmektedir. Bu iki değişken sağlanmadığında ise tutarlı bir tahmin yapılamamakta ya da yapılana kadar olay gerçekleşmiş olmaktadır. YSA bu noktada daha önceden eğitildiğinden tahmin süresini kısalttığı gibi daha doğru sonuçlara daha kısa sürede ulaşmak mümkündür.

Günümüzde hava tahmini için kullanılan birçok yaklaşım vardır.

[1] https://twitter.com/meteoroloji_twi/status/1408777140104044551

Matematiksel modelleme, istatistiksel modelleme ve yapay zekâ teknikleri bunlardan bazılarıdır. Araştırmacılar, mevcut atmosfer durumu ile olabilecek olan durum arasında bir doğrusal ilişki kurarak modelleri geliştirmişlerdir. Ancak pratikte atmosfer verileri doğrusal değildir. YSA doğrusal olmayan hava koşullarını belirli bir alan içinde işleyebilen ve tahminlerde bulunabilen bilgisayarlı bir sistem oluşturmak için etkili bir tekniktir (Fente, 2018). YSA, herhangi bir fiziksel sürecin doğrusal olmayan ve karmaşık temel özelliklerini yüksek bir doğruluk oranı ile yakalama yeteneğine sahiptirler (Fente, 2018).

Bu çalışmada; Yapay Sinir Ağları algoritmalarından ConvLSTM algoritması ile Hatay Plan Position Indicator (PPI) radar görüntülerinin tahmini yapılmıştır.

2. ÖNCEKİ ÇALIŞMALAR

Doğru ve tutarlı bir hava tahmini, özellikle yağış ve şiddetli hava olaylarının tahmini; insanların günlük yaşam aktivitelerinin yanı sıra tarım, inşaat, ekonomi gibi birçok alanda önemli bir etkiye sahiptir (Kim, 2017).

Kısa vadeli hava tahmini, yüksek çözünürlükte bir bölge ya da bir alan için 0-6 saat aralığını kapsayan meteorolojik tahmindir (Doviak, 1994). Daha genel bir ifade ile kısa vadeli hava tahmini, mevcut atmosfer durumunu gösteren hava parametreleri ile birlikte, anlık ve birkaç saat sonrası için bir bölge ya da bir alan için yapılan hava tahminidir (Sun, 2014). Meteoroloji radarı ile yapılan tahminler kısa vadeli hava tahminleri olup şiddetli hava olayları için erken uyarı işlevi görürler.

Günümüzde; sayısal hava tahmin modelleri, uzaktan algılama cihazları olan radar ve uydular, gelişmiş bilgisayar performansları, büyük veri ve yapay zekâ alanındaki gelişmeler hava tahmininin daha tutarlı olmasını sağlamaktadırlar.

Meteoroloji radarlarının temel amacı; o an havadaki yağışlı alanı bulmak, yağış türünü, yoğunluğunu, hareketini belirlemek ve kısa vadede yağış tahminini yapmak olsa da farklı birçok alanda da üzerinde çalışmaların yapılabileceği görülmüştür. McCarthy ve ark. (2019) tarafından yapılan çalışmada orman yangınlarının tespiti ve takibi ya da Lin ve ark. (2019) geçmiş radar görüntüleri ve yapay sinir ağları yardımı ile göçmen kuşların Amerika'da izledikleri yollar çıkarılmıştır. Benzer şekilde Nilsson ve ark. (2019) Avrupa radar ağını kullanarak kuşların gece göçleri ve göç yolları belirlemişlerdir.

Radar üzerinde yapılan çalışmaların büyük çoğunluğu istatiksel model içiermektedir. Örneğin Li ve ark. (2017) radar görüntü kalitesini istatiksel yöntemlerle doğrulayarak görüntülerin radar hatalarının belirlemesi ya da Fang ve ark. (2018) yaptıkları ve Z-R ilişkisinin isatistiği ve şiddetli yağmur hadisesinde radar hatalarının veri asimilasyonunu çalıştıkları makalede olduğu gibi. Fakat son çalışmalarda yapay zekânın kullanıldığı modeller artmaktadır. Bunun temel nedeni makine öğrenmesinin zamansal ve sıralı verilerin tahmininde daha tutarlı bir model

olmasından kaynaklanmaktadır. Zhang ve ark. (2020) yaptıkları diğer bir çalışmada ise birçok radar haritasını kullanarak ve sayısal model tahmininden yararlanarak Tiny-RainNet adını verdikleri bir model geliştirmişlerdir. Bu model çift yönlü Long Short-Term Memory (LSTM, Uzun-Kısa Vadeli Bellek) içermektedir. Aldıkları sonuçlar ile ConvLSTM modelinin LSTM, FC-LSTM gibi yapay öğrenme algoritmalarından daha iyi olduğu görülmüştür. Zhang ve ark. (2021) yaptıkları diğer bir çalışma ile önceki çalışmada kullandıkları modelden faydalanarak sürdürülebilir radar eko tahmini yapmışlardır. Bununla birlikte geliştirdikleri model teorik olarak kalmayıp bir uygulama alanı da bulmuştur.

Recurrent Neural Network (RNN, yenilenen / tekrarlanan yapay sinir ağları) mimarisi ile zaman bazlı ve sıralı vektörel verilerde oldukça iyi sonuçlar elde etmek mümkündür. RNN mimarisi ile ilgili ilk çalışmalar Tjandra ve ark. (1990) tarafından oluşturulmuştur. Hochreiter ve ark. (1997), RNN yapısından dolayı oluşan kaybolan gradyan problemini çözerek bu alana büyük katkılar sunmasından sonra ve bilgisayar donanım ve performans gelişmeleri, veri setlerinin artması ile bu alanda birçok çalışma yapıldı. Bu çalışmaların en önemlileri doğal dil işleme ve çeviri alanında bulunmaktadır. RNN ve LSTM algoritmaları kullanılarak, konuşma tanıma üzerine Hannun, A. ve ark. (2014) ile Amodei ve ark. (2016) çalışmışlardır. Yaptıkları çalışmalarda derin öğrenme ve LSTM algoritması ile konuşmayı tanıma işlemi gerçekleştirmişlerdir. Wu ve ark. (2016) ile Bahdanau ve ark. (2014) makine çevirisi üzerinden yoğunlaşarak tekrarlayan yapay sinir ağları ile makine çevirisi üzerine çalışmalar yapmışlardır.

Convolutional Neural Network (CNN) yani Evrişimli Sinir Ağı ile son yıllarda özellikle görüntü ve ses tanıma alanlarında çığır açan sonuçlar elde edilmiştir. Evrişimli Sinir Ağının en önemli özelliği yapay sinir ağlarındaki parametre sayısını azaltıp ayırt edici özelliklere odaklanabilmesidir (Albawi, 2017). Bu parametre eksiltme işlevi çok boyutlu verilerin işlenmesini kolaylaştırmıştır.

CNN ağları en çok çok boyutlu ve vektörel olmayan verilerin işlenmesinde kullanılır. Meteoroloji radar görüntüsü ise çok bandlı görüntü olduğundan CNN ile

işlenmesi daha uygundur. CNN algoritması Özellikle görüntü zenginleştirme işlemlerinde kullanılmıştır. Shi ve ark. (2018), geleneksel radar eko ekstrapolasyon yöntemlerinin uzun bir sınırlama süresi elde edilmesi zorluğundan dolayı radar verilerinin düşük kullanım oranlarını artırmak için dinamik CNN ağı ile görüntüleri ekstrapolasyon işlemine tabi tutmuşlardır.

Radar doğrudan yağış miktarı ölçmediğinden ve yağış miktarı radar görüntüleri kullanılarak bir denklem aracılığı ile hesaplandığından dolayı radar görüntülerinden yağış verisine ulaşmak en çok çalışılan konulardan biridir. Thorndahl ve ark. (2017) yaptıkları çalışmada radar görüntülerini kullanarak kentsel yağış miktarlarını hesaplamış ve tahminin yapmışlardır. Kullanılan verilerin her biri bir görüntü içerdiğinden ve bir zaman serisi olduğundan CNN ve LSTM ağlarının birleşimi birçok yöntemle kullanılmıştır.

Shi ve arkadaşlarının yaptığı bir çalışmada (2015), yerel bir gölge için kısa süreli yağış yoğunluğunu tahmin etmek için radar görüntülerini kullanarak, FC-LSTM algoritması ile tahmin etmeye çalışmışlardır. Shi ve arkdaşlarının yaptığı diğer bir çalışmada ise (2017), yağış miktarını belirledikleri 2015 yılındaki çalışmalarının modelini değiştirip ConvLSTM modeline dönüştürüp çalışmayı tekrarlamışlardır. Tekrarın sonucunda, ConvLSTM ağının mekânsal-zamansal korelasyonları daha iyi yakaladığını ve tutarlı bir şekilde FC-LSTM algoritmasından daha iyi performans gösterdiğini görmüşlerdir.

Kısa süreli tahmin yapılırken en büyük problemin zaman olduğu ve geleneksel matematik ve fizik model tabanlı tahmin modellerinin tahmini yapana kadar olayın gerçekleştiği sorununu göz önüne alarak Heye ve ark. (2017) ConvLSTM modeli ile daha yüksek alansal çözünürlükte ve kısa vade için tahmin yapmışlardır.

Agrawal ve ark. (2019) ise yüksek çözünürlükte radar görüntülerinden yararlanarak 1x1 km çözünürlükte ve bir saatlik yağış miktarı U-Net CNN ağı ile yağış tahmini gerçekleştirmişlerdir.

Niu ve arkadaşları (2020) çok kanallı radar görüntülerinden, ConvLSTM algoritması ile yağış tahmini yapmışlar. Oluşturdukları model, radar görüntülerini,

grid sıcaklığını, yağış miktarını ekleyip veri çok boyutlu hale getirilmiş ve yağış tahmini yapılmıştır.

Kim ve ark. (2017), üç boyutlu ve dört bandlı radar verileri kullanılarak gelecek 1 ve 2 saatlik yağış tahmini yapmışlardır. DeepRain olarak adlandırdıkları bu uygulamada da ConvLSTM algoritması kullanılmıştır. ConvLSTM algoritmasının, FC-LSTM algoritmasından en büyük farkı FC-LSTM'nin giriş olarak tek boyutlu veri analiz etmesidir. Radar, uydu gibi konum, zaman ve değer özelliklerini içeren veri olduklarından, uygulamalarda ConvLSTM algoritması daha iyi sonuçlar verdiği gözlenmiştir.

Kumar ve arkadaşları (2020) ConvLSTM temelli Convcast olarak adlandırdıkları, uydu verilerini kullanarak yağış tahminleri için yerleşik bir ConvLSTM tabanlı bir uygulama ile çok kanallı uydu görüntülerini kullanarak kısa vadeli yağış tahmini yapmışlardır. Yaptıkları bu çalışmada kayıt edilmiş on görüntü alınarak henüz kayıt edilmemiş uydu görüntüsü tahmin edilmiştir.

Radar verisi, veri tipi ve yapısı radar donanımını sağlayan firmaya göre değiştiğinden, bu görüntülerin okunması başka formatlara özellikle konum bilgisini da içeren başka formatlara çevrilebilmesi için programlama dillerinde kullanılan araçlar ve kütüphaneler geliştirilmiştir. Python poglama dili bunlardan biridir. Python programlama dilinde bu işlemler için yazılmış ve en çok kullanılan kütüphaneler, Py-ART (Helmus, 2006) ve Wradlib (Heistermann, 2013) kütüphaneleridir. Bu kütüphaneler aracılığı ile birçok radar ham veri formatı okunabilir, başka formatlara çevrilebilir ve bazı radar hataları giderilebilmektedir.

3. MATERYAL ve METOT

3.1. Materyal

3.1.1. Meteoroloji Radarı

Radar terimi, "RAdio Detection And Ranging" cümlesinin kısaltmasından meydana gelmektedir. "RAdio Detection And Ranging" kısaca radyo dalgaları ve bu dalgaların değişen değerleri anlamına gelmektedir (Bakır, 2018).

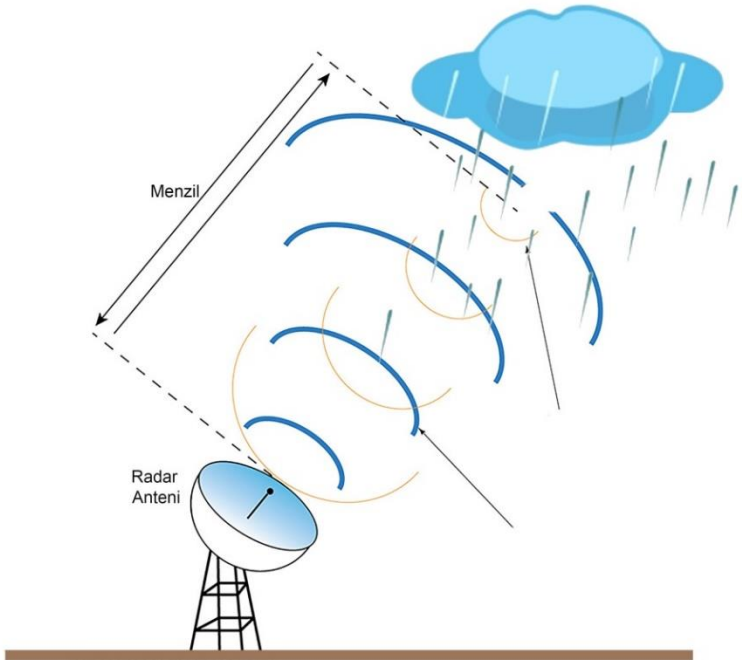
1840'lı yıllarda Christian Andreas, Doppler tarafından, gözlenen ışık frekansı ve ses dalgaları, kaynağın ve ölçüm cihazının göreceli hareketinden etkilendiğini keşfetmesi, 1860'lı yıllarda Michael Faraday'ın elektromanyetik dalgalarını keşfi ve James Clark Maxwell'in elektromanyetik dalga denklemlerini bulması ile radarın keşfine giden yol açıldı. Bu ve benzeri teorik keşifler ışığında 1880'li yıllarda Alman bilim insanı Heinrich Hertz tarafından radarın temelleri atıldı.

Radarlar, 1930'lu yıllarda günümüz radar kullanımı amacı ile kullanılmaya başlansa da, radar teknojisinin gelişiminin büyük çoğunluğu II. dünya savaşı ve sonrasında elde edildi. II. Dünya savaşında radarlar, düşman uçak ve gemilerinin konumlarının tespiti amacı kullanılıyorlardı. Savaş esnasında radar operatörleri radar ekranında hava olaylarının neden olduğu yankılar gördüler. Savaş sonrası çalışmalar ile radarlar meteoroloji alanında da kullanılmaya başlandı (Sokol, 2021). Doppler radar teknolojisinin gelişimi ile radarlardan sayısal veri elde edilmeye başlandı. Gelişen teknoloji ile radarlar meteoroloji biliminde tahmin ve gözlemin temel taşlarından biri haline geldiler. Meteoroloji radarları ile özellikle şiddetli yağışlar, dolu, su taşkınları ve selleri önceden belirlemek ve tahmin etmek mümkün olmuştur.

Meteoroloji radarları, anlık olarak havadaki genel durum ile yağışlı bölgeleri, bu alanların nerede ve radardan ne kadar uzakta olduğunu, hareketini, yönünü ve hızını, bulut içerisindeki yağış damlacıklarının büyüklüğünü ya da yoğunluğunu gösterirler. Elde edilen bu veriler yağış ve hava kütlesinin eğilimini tahmin etmek için kullanılırlar.

3.1.1.1. Meteoroloji Radar Çalışma Mantığı

Radarlar, donanımı aracılığı ile güçlü elektromanyetik dalga üretip, anten aracılığı ile hedefe gönderirler, daha sonra hedefe çarpan dalga dağılır, saçılır ya da yansır ve bir kısmı radara geri döner. Radar donanımı bu dönen miktarı alır ve üzerindeki yazılım ile gönderdiği ve aldığı dalga arasındaki farktan yola çıkarak bir sonuç elde eder. Şekil 3.1'de basit olarak radar çalışma mantığı gösterilmiştir.



Şekil 3.1. Radar Çalışma Mantığı

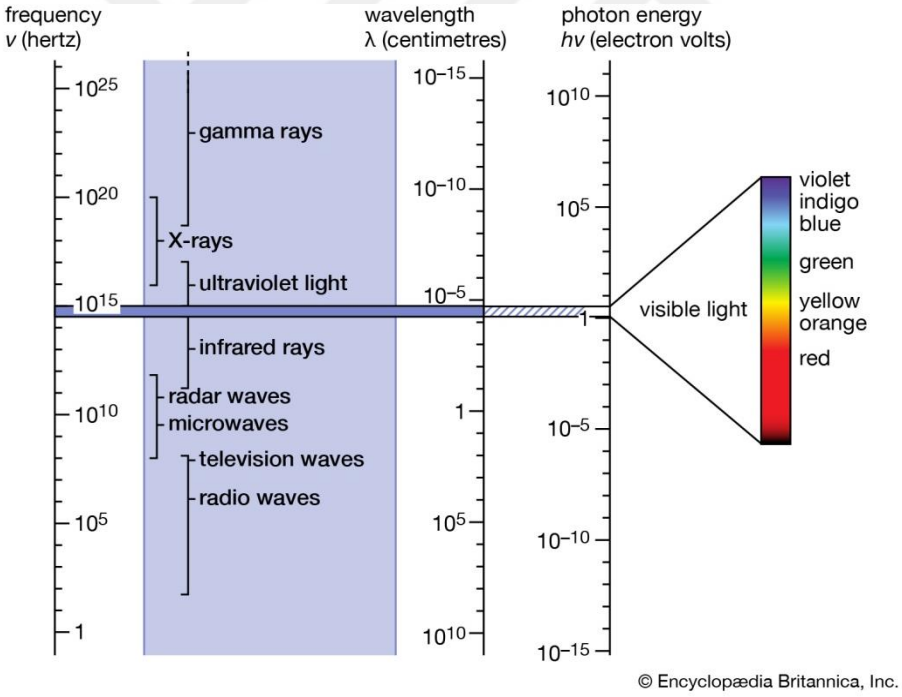
Bir uzaktan algılama cihazı, hedef nesne hakkında bilgi almak için kendi üzerinde bir enerji üretip bu işlemi tamamlıyorsa aktif, başka bir kaynaktan gelen enerjiyi kullanıyorsa yani kendi üzerinden enerji üretmiyorsa pasif bir uzaktan algılama cihazıdır.

Meteoroloji radarları, hedefe güçlü bir elektromanyetik dalga gönderip geri dönüşü dinlerler. Aktif sensöre sahip radarlar gönderilen ve alınan dalganın büyüklüğündeki ve polarizasyonundaki değişikliği algılayabilirler. Bu gönderilen ve

alınan sinyal arasındaki farklılıklardan yola çıkarak hedefin konumu ve büyüklüğü hesaplanabilir (Bakır, 2018).

3.1.1.2. Elektromanyetik Dalga Spektrumu

Elektromanyetik Dalga Spektrumu (EMS), radyo dalgaları, uzaktan kumandalardaki kızılötesi ışınlar, sterilizasyon cihazlarındaki morötesi ışınlar, röntgen filminde kullanılan X-Ray ışınları gibi bütün elektromanyetik dalgaları barındıran bir ölçektir. Elektromanyetik dalgalar temelde ışığı temsil eder ve periyodik olarak değişen elektrik ve manyetik alan oluştururlar. Değişen bu dalgaların frekansı ya da boyu ışığın fiziksel özelliğini dolayısı ile EMS üzerindeki yerini belirler.



Şekil 3.2. Elektro Manyetik Tayf [2]

[2] https://www.britannica.com/science/electromagnetic-spectrum

3.1.1.3. Radar Çeşitleri

Radarlar, sinyal alma ve iletim tipi, kullanım amacı ya da alanı, çalışma frekansı, sinyal yayma tipi, polarizasyon tipi gibi birçok ana başlık altında sınıflandırmanın yapılması mümkündür.

Meteoroloji radarlar ise elektromanyetik dalganın frekans bandına göre isimlendirilirler. Radar frekans bandları ilk olarak 1976 yılında tanımlanmış olup, teknolojik gelişmelere ve ihtiyaçlara göre bu standartlar değiştirilmiştir. Son hali ise Institute of Electrical and Electronics Engineers (Elektrik ve Elektronik Mühendisleri Enstitüsü, IEEE) tarafından 2019 yılında güncelleştirilip yayınlanmıştır. Çizelge 3.1'de IEEE'nin yapmış olduğu adlandırma görülmektedir. NATO ve ITU tarafından yapılan farklı adlandırmalar da mevcuttur.

Çizelge 3.1. IEEE 521-2019 standardına göre meteoroloji radar çeşitleri ve frekans harf kısaltmaları (IEEE, 2020).

Band	Nominal Frekans Aralığı	Dalga Boyu
L	1 to 2 GHz	30 – 15 cm
S	2 to 4 GHz	15 – 7.5 cm
C	4 to 8 GHz	7.5 – 3.75 cm
X	8 to 12 GHz	3.75 – 2.4 cm

3.1.1.4. Radar Hataları

Radarlar, uzaktan algılama ile hedef nesne hakkında veri elde ederken, elde ettiği veri sadece hedefe ait olmaz, hedef dışında nesnelerden de yankı aldığından dolayı veri her zaman hedef nesne özelliklerini içermez. Örneğin askeri amaçlar için kullanılan radarlarda bir süre sonra düşman unsurlarından başka hedefler görülmüş ve sonra bu görüntülerin yağışlı bölgeyi temsil ettiği fark edilmiştir. Bu durum II. Dünya savaşı sonrasında meteoroloji radarlarının doğmasına neden olmuştur (Sokol, 2021).

Meteoroloji radarları, atmosferdeki meteorolojik olaylar özellikle yağış, dolu, bulutluluk gibi meteorolojik parametreler hakkında bilgiler sağlarlar. Ancak radarların çalışma mantığı, donanımları, konumları, çevresindeki topografya gibi

nedenlerden dolayı yalnızca bu meteorolojik parametreleri ölçmezler. Bazen gözlemledikleri bu yağış olmayan değerler, radar verilerinde hatalara neden olur. Hatalar ise tutarlığı daha düşük bir tahmin ya da yanlış bir gözleme sebebiyet verir.

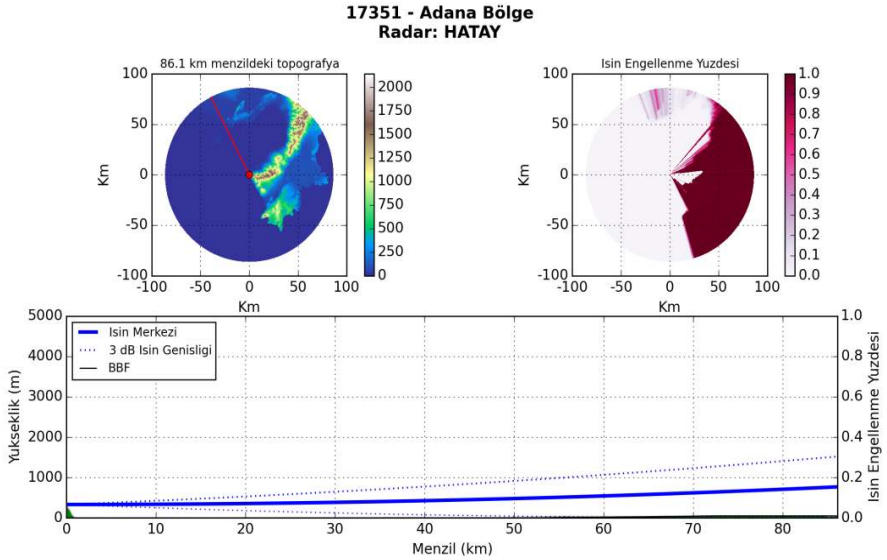
Radar verisinin hatalı olmasına neden olabilecek birçok neden bulunmaktadır. Bu hatalı veri oluşmasının başlıca nedenleri: Z-R bağıntısından kaynaklı sapma, ışın engellemesi (beam blockage), yerkürenin şekli (earth curvative), yağışın düşey dağılımı profili, sinyal zayıflaması (attenuation), elektromanyetik girişim, zemin dağınıklığı (ground clutter), anormal yayılım (anomalous propagation), menzil belirsizliği (range ambiguity), parlak bant etkisi (bright band effect), radar denkleminden kaynaklı sapma, radar donanımından kaynaklı sapma.

Bu bölümde radar hatalarından önemlilikleri ve yazılımsal olarak düzeltilebilecek olanlar incelenmiştir.

Işın Engellemesi (Beam Blockage): Dağlık ya da farklı yükseltilere sahip yerlerde çalışan meteoroloji radarları, genellikle çevrelerini kaplayan yükseltilerden ışın blokajına maruz kalmaktadırlar. Yükseltiler dolayısı ile radar ışınları yükseltiler tarafından kesilir (Chumchean, 2003: Fabry, 2015: Villarini, 2009). Kesilen ve yükseltiye çarpıp geri dönen ışınlar radar görüntüsünde hatalı veri oluşturur.

Radarlar en doğru veriyi düşük anten ve yükseklik değerlerinden alırlar. Dağlık bir coğrafyaya sahip ülkemizde, daha az ışın engellemesine maruz kalmak için radarlar yüksek yerlere kurulmuşlardır.

Bu çalışmada kullanılan Hatay radarının; Adana için ışın engellemesi görüntüsü, MGM tarafından geliştirilen Türkiye Radar Görünürlük Analizi İnteraktif Uygulaması ile oluşturulmuştur. İlgili görüntü Şekil 3.3'te verilmiştir. Grafiğin skalası 0-1 aralığında olup, 0 ışının herhangi bir engel ile karşılaşmadığı 1 ise ışının engel ile karşılaşıp tamamen blokaja uğradığı anlamını taşımaktadır. Bir altındaki grafik iseradar ile seçilen noktanın arasındaki alanın dikey kesitidir.



Şekil 3.3. Hatay Radarı Adana Meteoroloji Bölge Müdürlüğü için görünürlük analizi (MGM, 2020).

Yerkürenin Şekli (Earth Curvative): Radar ışınları doğrusal olarak ilerlediklerinden ve dünyanın ise düz olmayıp geotit bir yapısı olmasından dolayı, ışınlar radardan uzaklaşıldıkça yer ile arasındaki mesafe artar, dolayısı ile radar kendisine yakın yerde taradığı hacim ile menzil arttıkça taradığı hacim arasında fark olur. Bu da radara yakın yerdeki bir piksel ile radardan uzak bir piksel verisi aynı değerde olsa bile her zaman aynı şeyi ifade etmeyeceği anlamına gelir. Radardan uzaklaştıkça piksel verisi daha az güvenilir olur (Villarini,2009: Bakır, 2018: WMO, 2014: WMO,2017)

Soğurma (Attenuation): Radar elektromanyetik dalgaları atmosferde hedeflerine doğru ilelerken, içinden geçtikleri atmosfer yapısından etkilenirler, aynı şekilde hedeften geri yansıyan dalgalar radara ulaşırken aynı süreçten geçtiklerinden zayıflarlar. Dolayısı ile radara dönmesi gereken ya da hedefe gitmesi gereken elektromanyetik dalga saçılmaya uğradığından radar verisinde hatalara neden olur. (Chumchean, 2013: Villarini, 2009: WMO, 2014)

Zemin Dağınıklığı (Ground Clutter): Radar çevresinde tepe, bina, ağaç gibi

engeller var ise radar elektromanyetik dalgaları bu engellere çarpıp geri dönebilecekleri gibi yansıyıp yayılabilirler. Bu durum radar değerlerinde yanıltıcı reflektivite değerlerinin oluşmasına neden olur.

Zemin dağınıklığı hatasını aşmak için radar verisine ait her bir piksel için hız-kazanç Gaussian dağılımı yapılıp, bu etkinin var olduğu piksele ait veriler ayıklanabilir. (Chumchean, 2013: Villarini, 2009: WMO,2014) Tamamen bulutsuz bir güne ait radar taraması yapılarak zemin dağınıklığı pikselleri tespit edilip maskelenebilirler (Villarini, 2009).

Anormal Yayılım (Anomalous Propagation): Gönderilen elektromanyetik dalga ışınındaki yayılma yönünde oluşan bozulmadır. Radar ışınının yeryüzüne doğru kırılması ile birlikte bükülmesi ve yeryüzüne çarparak radara doğru geri yansıyarak yayılması durumudur

Radar tarafından gözlemlenen bir alanda sıcaklık ve su buharı dikey dağılımı; iklim koşullarına, atmosferik dolaşıma ve taranan alanın özelliklerine bağlıdır (Mesnard, 2010). Değişen bu dağılım bazen radar tarafından gönderilen dalganın kırılmasına dolayısı ile radar görüntüsünde yağış nedeni ile oluşmayan bir eko oluşmasına neden olmaktadır.

Menzil Belirsizliği (Range Ambiguity): Radardan gönderilen elektromanyetik dalgalar ışık hızıyla hareket ederken, darbe radarının uzaktaki nesneleri algılaması için belirli bir alma süresine ihtiyacı vardır. Bu alma süresine dalganın tekrar aralığı denir ve darbe ve bekleme süresini içerir ve radar menzilini belirler (Xu, 2018). Darbeler arası süre yeterince uzunsa, bu bir sorun teşkil etmez. Darbeler arası süre kısaltıldığında, bir önceki darbeye kadar geçen süre, geçen gerçek süreden daha kısadır ve yanlış (belirsiz) daha kısa bir aralık verir. Yani ikinci dalgada halen birinci dalga dinleniliyor olabilir (Bakır, 2018). Bu algılanan ve gerçekte uzakta olan ekonun yakınmış gibi değerlendirilmesine neden olarak bir menzil belirsizliğine yol açar.

Parlak Bant Etkisi (Bright Band Effect): Atmosferde bulunan su buharının katı ve sıvı halde farklı dielektrik katsayısına sahip olmalarından dolayı farklı

büyüklükte enerji yansıtma oranlarına sahiptirler. Suyun sıvı hali katı halinden beş kat daha fazla enerji yansıtır. Hal değişimin olduğu 0 derece seviyelerinde radar verisi üzerinde reflektivite değerlerinde ani bir artış meydana getirir. Bu parlak band etkisi hal değişimi tamamlanıncaya kadar devam eder (Fabry 2015: Fukao 2014).

3.1.2. Radar Ürünleri

3.1.2.1. Radar Ürünlerinin Elde Edilmesi

Radar ürünlerinin elde edilmesi, radar donanımından alınan sinyallerin işlenmesi ile başlar, bu aşamada veriler sadece reflective ve velocity değerlerinden oluşan ham veridir. Bu veriler işlenebilecekleri bir merkez bilgisayarlara aktarılır. Radar yazılımları aracılığı ile ham verilerden temel ve kompozit radar ürünler elde edilir. Elde edilen veriler son kullanıcıya radar sunum yazılımları aracılığı ile aktarılır.

3.1.2.2. Radar Parametreleri

Temel radar parametreleri: Reflektivite (Z), Hız (mm/saat), Spektral Genişlik (m/s), Diferansiyel Reflektivite (dBZ) ve Yağış Miktarıdır (mm).

Reflektivite, kısaca radardan gönderilen sinyalin geri saçılma sonucu radar anteni tarafından toplanması sonucu bu gönderilen ve alınan sinyal arasındaki farkın basit bir gösterimidir. Daha büyük hedefler daha fazla sinyal geri yansıtacaklarından daha yüksek dBZ değerleri elde edilir.

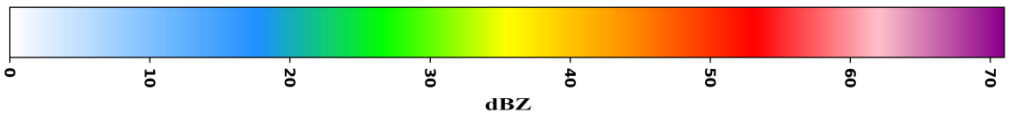
Yağışa neden olmayan ya da çok az yağış meydana getiren bulutların reflektivite değeri zayıftır. Z değeri, 0.001-40.000.000 arasındadır (dBZ: -30 (sis) ile +76 dBZ (büyük dolu) arası). Radar görüntüleri yorumlanırken düşük miktardaki yağışlarla ilgilenilmez dolayısıyla yağışsız bu bölgeler genellikle gösterilmez. kısaca reflektivite renk skalası negatif değerleri içermez (MGM, 2020).

Çizelge 3.2'de reflektivite değerlerine karşılık gelen yağış tipleri ve diğer eko kaynakları gösterilmiştir.

Çizelge 3.2. Çeşitli eko kaynaklarına karşılık gelen radar reflektivite faktörü değerleri.[3]

Yağış / Hadise	dBZ
Su içeren fakat yağıs vermeyen sis - bulutlar	< 0 dBZ
Buz parçacıkları içeren bulutlar	20 dBZ'ye kadar
Çisenti	0-20 dBZ
Hafif yağmur	10-30 dBZ
Şiddetli yağmur - hafif sağanak	30-45 dBZ
Şiddetli sağanak	>40-65 dBZ'ye kadar
Dolu	Donma seviyesi üzerinde dBZ >45 Eger dBZ > 55 ise tum yüksekliklerde mümkün
Kar	35 dBZ'ye kadar
Duman - Toz - Böcek (Yerden 2km yüksekliğe kadar)	10 dBZ'ye kadar
Clutter (Yeryüzünden, binalardan, ağaçlardan, su yüzeylerinden v.b. olan istenmeyen ekolar)	Filtrasyon yapılmadiginda 80 dBZ'ye kadar
Kuşlar	20 dBZ'ye kadar

dBZ değerleri radar ekranında renkler ile gösterilmektedirler. Renkler, desibel (dBZ) değerleriyle ifade edilen radara geri dönen enerjinin gücünü temsil etmektedir. Renk skalası kullanılan radar arayüzüne, radar ürününe göre değişebilmektedir. Şekil 3.4'te PPI görüntüler için kullanılan örnek renk skalası verilmiştir.



Şekil 3.4. dBZ renk skalası.

Hız ise radar tarama sahasında tespit edilen hedeflerin hızını temsil eder. Bu ölçüm doppler ile yapılır. Doppler radarları yalnızca bir hedeften alınan gücü

[3] https://hezarfen.mgm.gov.tr/Helimet/Helimet%20Klavuz.pdf

algılayıp ölçmekle kalmaz, aynı zamanda hedefin radardan uzaklaşan ya da yaklaşma hareketini de ölçer. Buna "Radyal Hız" denir. Radyal hız, hedefin Doppler frekans kaymasından belirlenir. Hareket eden hedefler, dönen sinyalin frekansını değiştirir. Bu frekans kayması daha sonra rüzgâr hızını belirlemek için kullanılır (WMO, 2006).

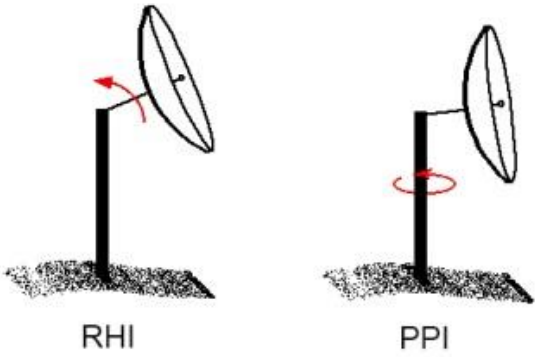
Spektral genişlik, örnek radar hacmi içerisindeki hız dağılımını tanımlayan meteorolojik parametrelerin bir fonksiyonudur. Radar, bir piksel için görüntülenen tek bir ortalama radyal hız üretmek için bir hacim örneği ile ayrı ayrı radyal hızların ortalamasını alır. Kayma ve türbülansın bir piksel içinde küçük olduğu bir durumda, spektrum genişliği küçük olacaktır. Bir piksel içinde kayma ve radyal hızın büyük olduğu bir durumda, spektrum genişliği büyük olacaktır. Spektrum genişliğini tanımlamanın teknik bir yolu, tek bir piksel içindeki hız dağılımının standart sapmasıdır.

Diferansiyel reflektivite, hedeften yansıyan yatay ve dikey reflektivite değerlerinin birbirlerine oranıdır. Bu oran sıfıra ne kadar yakınsa hedef şekilsel olarak küreye o kadar yakındır. ZDR değeri yağış türlerinin sınıflandırılmasında kullanılır.

Radarlar doğrudan yağış miktarını ölçmezler. Radar ölçümleri sonucunda çıkan yağış ürünleri; Z ve R fonksiyonu ile elde edilir.

3.1.2.3. Temel Radar Ürünleri

Radarlar verileri; kullanılan tarama yöntemlerine göre elde ederler. Radarlar Plan Position Indicator (PPI), Range Height Indicator (RHI) olmak üzere iki tür tarama yapmaktadırlar. PPI taramada radar yükseklik açısını sabit tutar ancak azimuth açısını değiştirir. Radar kendi etrafında 360 derece dönüş yapıp tarama yapabileceği gibi belli bir aralıkta sektör taraması da yapabilir. RHI tarama yaparken, radar azimut açısını sabit tutar ancak yükseklik açısını değiştirerek tarama yapar. Bu taramalar sonucunda radar ürünleri oluşturulur. PPI ve RHI tarama yöntemleri Şekil 3.5'te gösterilmiştir.



Şekil 3.5. PPI ve RHI tarama.[4]

PPI; radar anteni belirli bir yükseklik açısında sabit tutularak yatayda tarama yapılarak hedefe sinyal gönderilir. Alınan sinyal ile hedefin gerçek koordinatı ve varsa yağışlı bölgeler belirlenir. PPI ürün hedefin konumu ve şiddeti ile ilgili bir öngörü ürünüdür.

Maxsimum Display (MAX); bulut içeriği, yoğunluğu ve yüksekliği hakkında bilgi sağlar. Aynı noktayı gösteren piksellerin maksimum değerlerini gösterir.

Constant Altitude PPI (CAPPI); antenin belli bir yükseklik açısından başlayarak, diğer bir yükseklik açısına kadar PPI tarama yapılarak elde edilen görüntüdür.

RAINN ürünleri ise belirlenen N saat için toplam yağış miktarını gösterir. TOPS ürünü reflektivite eşik değerinin üzerinden belirlenen bir değerin en yüksek olduğu yükseklik seviyesini gösterir. WIND ürünü ise yatay rüzgâr vektörlerini içerir. VIL (Vertical Integrated Liquid) ürünü alt ve üst seviyesi tanımlanan bir tabaka içerisinde bulunan su içeriğinin anlık hacimsel değerini gösterir (MGM, 2020).

Bu tez çalışmasında PPI görüntüsü kullanılmıştır.

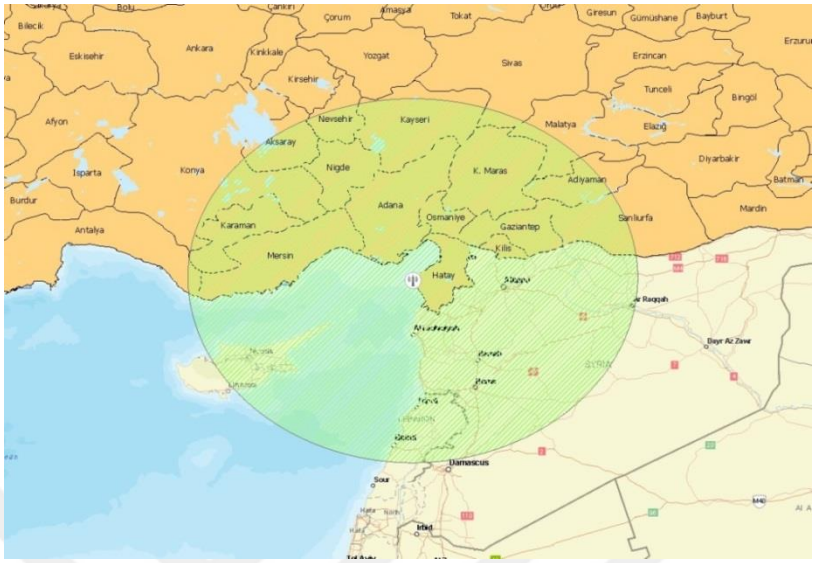
[4] http://ww2010.atmos.uiuc.edu/(Gh)/guides/rs/rad/basics/cnmod.rxml

3.1.3. Meteoroloji Hatay Radarı ve Çalışma Alanı

Meteoroloji radar ağı kurulması kapsamında Hatay, Arsuz ilçesi Işıklı köyü yakınlarında Hava Gediği Tepesine radar kurulmuş olup 2011 yılından itibaren işletmeye alınmıştır. Meteoroloji Hatay Radarının denizden yüksekliği 304 metredir. Meteoroloji Hatay radar anteni ve yerleşkesi Şekil 3.6'de, Meteoroloji Hatay radarı C-Band Polarimetric Doppler radarıdır, PPI görüntü tarama alanı da Şekil 3.7'da verilmiştir.

Şekil 3.6. Hatay meteoroloji radar anteni ve yerleşkesi.[5]

[5] https://www.mgm.gov.tr/genel/meteorolojiradarlari.aspx?s=radaragi



Şekil 3.7. Hatay meteoroloji radarı konumu ve çalışma alanı.

3.1.4. Yapay Öğrenme

Türk Dil Kurumuna göre öğrenme; belirlenmiş problemler karşısında cevap, davranış ya da tepki oluşturma süreci olup, karşılaşılan benzer ya da başka durumlar için var olan davranışları değiştirme ve yeni davranış oluşturma süreci olarak tanımlanabilir. Yapay öğrenme ise canlılara has bu sorun çözme akışının algoritmalarla makinelere yaptırılmasıdır.

Makine öğrenmesi algoritmaları, bir ürünün performansını etiketlenmiş verilerle ya da geçmiş deneyimlerle optimize etmeyi amaçlamaktadır. Makine öğrenimi gerçekleştirebilecek birçok algoritma vardır. Bu algoritmalar genel olarak üç ana başlıkta toplanır. Bunlar; denetimli öğrenme, denetimsiz öğrenme ve pekiştirmeli öğrenme algoritmalarıdır

3.1.4.1. Denetimli Öğrenme

Öğrenme; insanlarda deneyimlerden, makine öğrenme algoritmalarında ise veriden gerçekleşir. Bir yapay öğrenme algoritmasına verilen veri etiketli, yani hangi

verinin hangi çıktısı olduğu biliniyorsa ve öğrenme algoritması bu bilinen girdi ve bilinen çıktı verisi arasındaki bağıntılar ile öğrenmeyi gerçekleştiriyorsa bu bir denetimli öğrenmedir (Jiang, 2020).

Örneğin radar görüntülerinden yağış miktarı tahmini yapılmak istenirse; girdi olarak meteoroloji radar görüntüsü ve bunun çıktısı olarak yer meteoroloji istasyonlarında ölçülen bir yağış verisi bulunur. Bu etiketlenmiş bir veri olup her görüntü, o görüntüye ait yağış miktarı çıktısı ile etiketlenmiştir.

Denetimli öğrenmede; algoritma etiketlenmiş girdi verisini bir bağıntı ile yine bilinen bir çıktı verisine bağlar. Böylece girdi verisinin bilinen çıktıya ulaştıracak ağırlık değerlerini elde eder. Bu ağırlık değerlerini test verisi üzerinden kontrol edip gerekiyorsa düzeltir. Etiketini bilmediği bir veri içinde bulduğu ağırlık değerlerinden tahmin yapmaya çalışır. Bilinen bir girdi verisi seti ve verilere bilinen yanıtları alır, ardından yeni verilere yanıt için makul tahminler oluşturmak üzere modeli eğitir.

Denetimli öğrenme; bir modelin etiketli bir veri kümesi üzerinde eğitildiği, yani hem giriş hem de çıkış parametrelerine sahip bir türdür. Makine öğrenmesinde sıklıkla kullanılır.

3.1.4.2. Denetimsiz Öğrenme

Denetimsiz öğrenme veriler etiketlenmemiştir. Yani veriler sebep-sonuç ya da girdi-çıktı şeklinde etiketlenmezler. Algoritma veri içerisinde var olan ya da olabilecek bağlantıları, kalıpları matematiksel ya da istatistiksel ilişkiler bularak öğrenir. Denetimsiz öğrenme algoritmaları çoğunlukla sınıflamada kullanılırlar.

3.1.4.3. Pekiştirmeli Öğrenme

Bazı problemlerde, bir eylemin sonucu tek başına önemli değildir. Başarılı sonuç ancak birkaç eylemin sıralanması ile oluşur. Ara bir durumda en iyi eylemi tanımlamak bazen yanlış olur. Bir eylem ancak iyi prensiplerin bir parçası ise anlam taşır (Apaydın, 2012).

Pekiştirmeli öğrenmede, algoritma deneme yanılma yolu ile hareket ederek en

fazla ödül topladığı eylemleri yapma eğilimde olur. Yanlış seçimler ise cezalandırılır.

Pekiştirmeli öğrenme, veri etiketlenmediğinden ve uygun eylemlerin dış müdahale ile düzeltilememesi ile denetimli öğrenmeden ayrılır. Pekiştirmeli öğrenme; içeriği ya da özellikleri tam bilinmeyen bir veri kümesinde keşif ile var olan bilgiden yararlanma arasında bir denge kurma modelidir (Kaelbling, 1996).

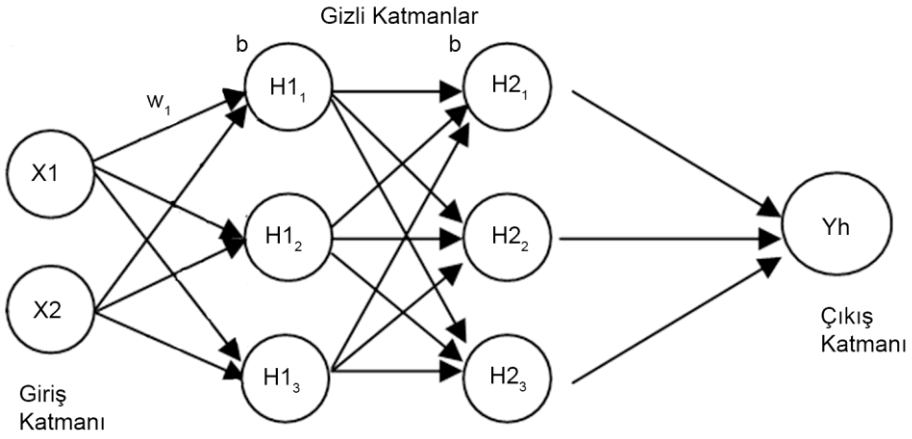
3.1.5. Yapay Sinir Ağları

Yapay sinir ağları; bir bilgi işleme yöntemi olup, beynin öğrenme şeklini matematiksel olarak modellenmesini sağlayan yöntemler bütünüdür (Joselin, 2018). YSA öğrenme, öğrendiklerini kaydetme, veriden yeni bilgiler üretme ve veriler arasında ilişkiler oluşturabilme yeteneklerine sahiptirler. YSA günümüzde birçok alanda kullanılsa da en yaygın kullanıldığı yerler; görüntü tanıma, veri analizi, kontrol, kümeleme ve sınıflandırma problemlerinin çözümünde kullanılır.

3.1.5.1. Feedforward Neural Network

İleri beslemeli yapay sinir ağları, en basit YSA olup veri ileri doğru gizli katman/katmanlardan geçirilip çıkışa aktarılır (LeCun, 2015).

Şekil 3.8'de bir ileri beslemeli yapay sinir ağı modeli gösterilmiştir. Şekilde görüldüğü üzere iki giriş ve bir çıkış değerine sahip, iki gizli katmandan oluşan basit bir ileri beslemeli yapay sinir ağı yapısı bulunmaktadır. Şekilde giriş değerleri X1 ve X2 çıkış ise Yh ile gösterilmiştir. Her gizli katmanda bir önceki gizli katmandan ya da girişten gelen veri alınıp, W_x ağırlık değerleri ile çarpıldıktan sonra b_x ile toplanır ve aktivasyon fonksiyonundan geçirildikten sonra bir sonraki katmana aktarılır. Yh değeri bulunduktan sonra olması gereken değer ile Yh karşılaştırılır. Geriye yayılım algoritması kullanılarak ağırlık ve bias değerleri güncellenir.



Şekil 3.8. Feedforward Neural Network yapısı.

Birçok derin öğrenme uygulaması, örneğin bir görüntüyü sabit boyutlu bir giriş değeri olarak alır, veriyi ileri doğru işleyerek, sabit boyutlu bir çıktıya eşlemeyi öğrenen ileri beslemeli katmanlı sinir ağı mimarilerini kullanır. Bir katmandan diğerine geçmek için, bir dizi birim, önceki katmandan gelen girdilerinin ağırlıklı toplamını hesaplar ve sonucu doğrusal olmayan bir işlevden geçirir.

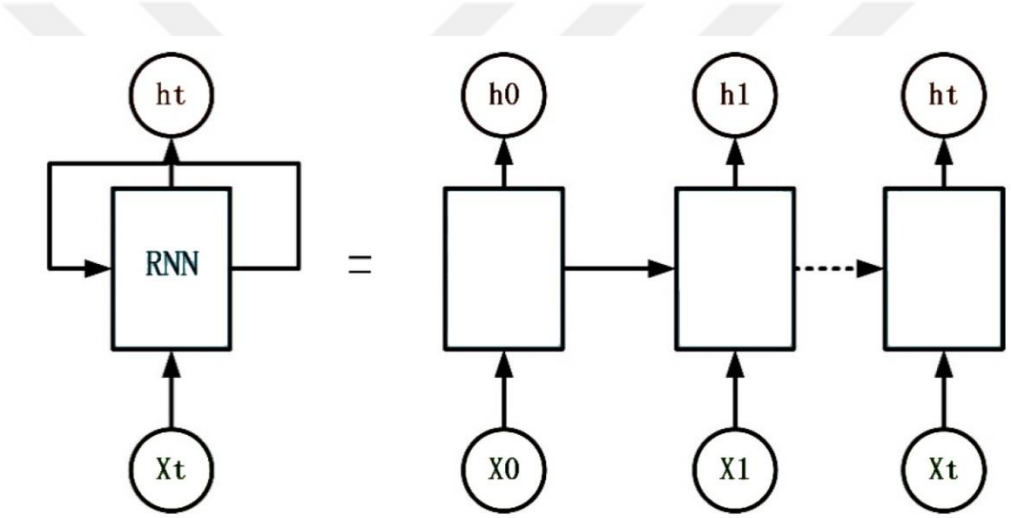
Doğrusal olmayan fonksiyon, katmanın herhangi bir çıkışından gelen değeri bir aktivasyon fonksiyonu ile bir sonraki çıkışa verir. Problem türüne ya da elde edilmek istenen çıkışa göre aktivasyon fonksiyonu kullanılır.

3.1.5.2. Recurrent Neural Networks

RNN, geleneksel çok katmanlı yapay sinir ağlarından farklı olarak bilginin kalıcı olmasını sağlayan bir yapıya sahiptir. Geleneksel sinir ağlarında girişler birbirlerinden bağımsız iken RNN yapısında bir girdinin çıkışı, sıradaki veriye girdi olarak verilir bu ağda bir bağımlılık yaratır.

Konuşma ve dil, zaman serileri gibi sıralı girdileri içeren görevler için, genellikle RNN'leri kullanmak daha başarılı sonuçlar verir (LeCun, 2015; Dipietro 2020). Çünkü sıralı ya da zamana bağlı değeri bulunan veriler birbirlerine bağlıdır. Bir cümlede; sıradaki seçilecek kelime, önceki kelime ya da kelimelerden etkilenir.

Hava sıcaklığı bir önceki günlerin sıcaklıklarından ya da başka meteorolojik parametrelerden etkilenir. Bu tür sıralı yapılarda veriyi birbirinden bağımsız işlemek probleme çözüm olmaz. Veri RNN'de işlenirken bir önceki girdi ya da girdilerin kendisinden sonra gelecek olan girdiyi etkiler. RNN'ler, her seferinde bir girdi olarak bir veri dizisini işler, gizli birimlerinde dizinin tüm geçmiş öğelerinin geçmişi hakkında örtük olarak bilgi içeren bir durum vektörü tutar (Zhao, 2017). Şekil 3.9'da bir gecikme hattına sahip ve iki zaman adımı için zaman alanında açılmış temel bir RNN mimarisini göstermektedir.



Şekil 3.9. RNN Mimarisi

3.1.5.3. Kaybolan Gradyan Problemi (Vanishing Gradient Problem)

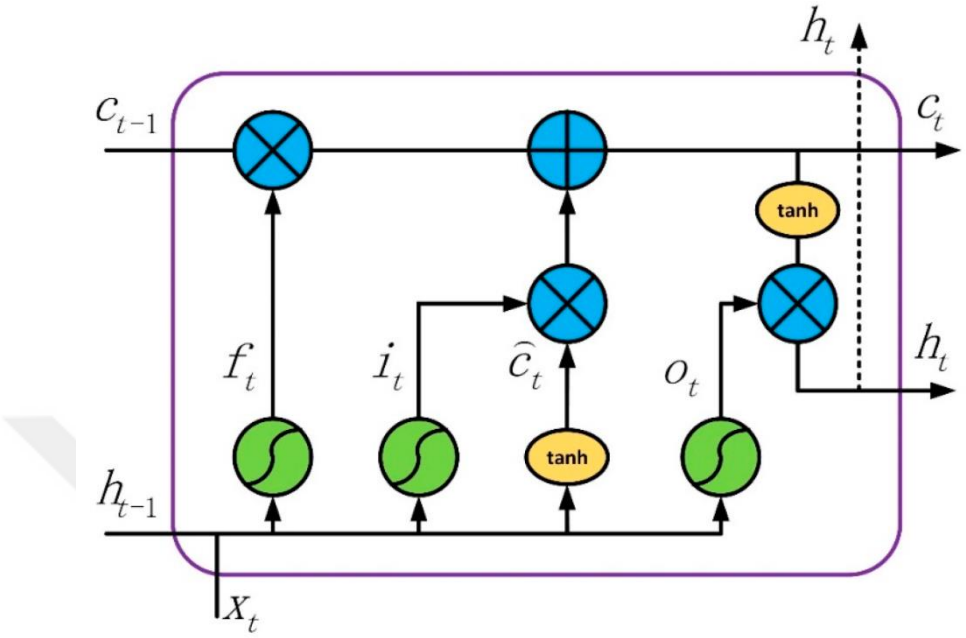
Derin yapay sinir ağlarında veri ileri doğru bir katmandan diğerine geçerken aktivasyon fonksiyonu ile çarpılıp bir sonraki katmana iletilir. Geri yayılım algoritması gereği çıkış katmanında oluşan değer olması gereken değer ile karşılaştırılıp son katmandan ilk katmana doğru türevlenir. Böylece ağırlık değerleri güncellenir. Ama geri yayılımda kısmi türev alındığından, son katmandan ilk katmanlara doğru gelindiğinde ağırlık matrisi ile çarpılacak değerler 0 ya da 0' a çok yaklaşır. 0'a her yaklaşma ağırlık değerlerinin o kadar az güncellenmesini sağlar.

Tam 0 değerinde ise ağırlıklar güncellenmez. Dolayısıyla ağ güncellenmediğinden öğrenemez. İstenilen şey ağırlıkların güncellenmesi ve ilk katman çıkışlarının ağrılıklarının azaltılmaması olduğundan bu da bir kaybolan gradyan problemi oluşturur. Bu problem aşmak için aktivasyon fonksiyonu değiştirilebilir. Türevlendikçe değeri sıfıra yaklaşan Doğrusal olmayan aktivasyon fonksiyonları yerine doğrusal aktivasyon fonksiyonları seçilebilir. Örneğin sigmoid fonksiyonu yerine Relu ya da Tanh fonksiyonu kullanılabilir. RNN yapay sinir ağlarında ise LSTM yapay sinir ağı modeline geçilebilir. (Hochreiter,1997: Roodschild, 2020)

3.1.5.4. Long Short Term Memory (LSTM)

LSTM ağları, dizi tahmin problemlerinde sıra bağımlılığını öğrenebilen bir tekrarlayan YSA türüdür. RNN ağlarının eğitilmesi, kaybolan gradyan problemi dolayısı zorluklar barındırır ve veri sayısı arttıkça doğru sonuçlara ulaşması zorlaşır. Bu zorluğu aşmak için Hochreiter ve ark tarafından 1997 yılında LSTM çözümünü ortaya atmışlardır. RNN ağlarının eğitilmesinde yaşanan problemler LSTM ile giderilmiştir. LSTM, hafıza geçişli mekanizması ile uzun vadeli bağımlılıkları öğrenebildiği için sıralı veya zaman serisi problemlerde oldukça fazla kullanılmaktadır (Ahmadi, 2019).

LSTM, RNN ağ yapısına benzer bir akış kontrolüne sahip olsa da temel farklılık ağ içinde yapılan işlemlerdir. RNN bir adet aktivasyon fonksiyonu içeren katmandan oluşurken, LSTM iletişim halinde olan 3 farklı katman içerir. Şekil 3.10'da temel LSTM blok şeması verilmiştir.



Şekil 3.10. LSTM mimarisi (Fang, 2020)

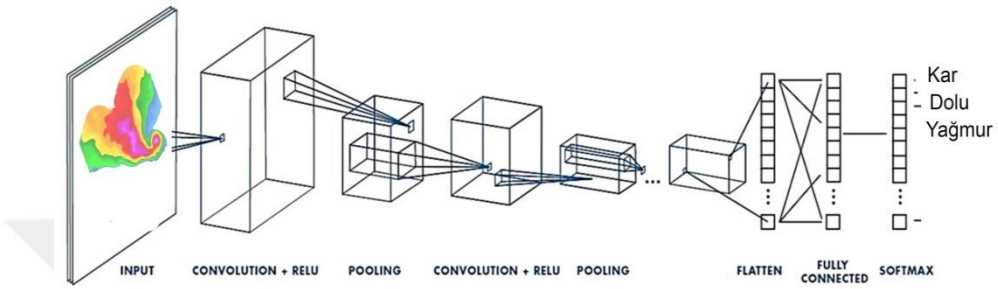
Şekil LSTM 3.7'de gösterilen şemaya göre kapılar mavi ile boyanmıştır. Yeşil ve sarı alanlar ise aktivasyon fonksiyonlarını temsil etmektedir. X_t giriş verisi olup, C_t hücrenin durum kapısıdır. Mavi dairelerle belirtilen alanlar ise işlemleri belirtmektedirler. LSTM, uzun vadeli bağımlıkları öğrenebilen özel bir RNN türüdür.

LSTM, konuşma ve ses tanıma, doğal dil işleme, duygu analizi, yapay müzik oluşturma, sinyal analizi, insan hareketi tanıma, resim yazısı oluşturma, hava tahmini ve video sahne tahmini gibi birçok alanda kullanılmaktadır.

3.1.5.5. Convolutional Neural Network

Evrişimli Sinir Ağı, görüntü tanıma, görüntü işleme ile ilgili çeşitli alanlarda son on yılda çığır açan çalışmalar yapılmıştır. CNN'lerin en faydalı yönü, görüntü işleme ses tanıma gibi çalışmalarda, ağdaki parametre sayısını azaltmasıdır. CNN modelindeki bu kolaylık araştırmacıların YSA ile çözülmesi mümkün olmayan

görevlerin başarılması sağlanmıştır (Albawi, 2017). Örneğin görüntü tanıma, el yazısı tanıma işlemleri gibi problemler CNN ile çözülmüşlerdir. Şekil 3.11'de temel CNN yapısı gösterilmiştir.

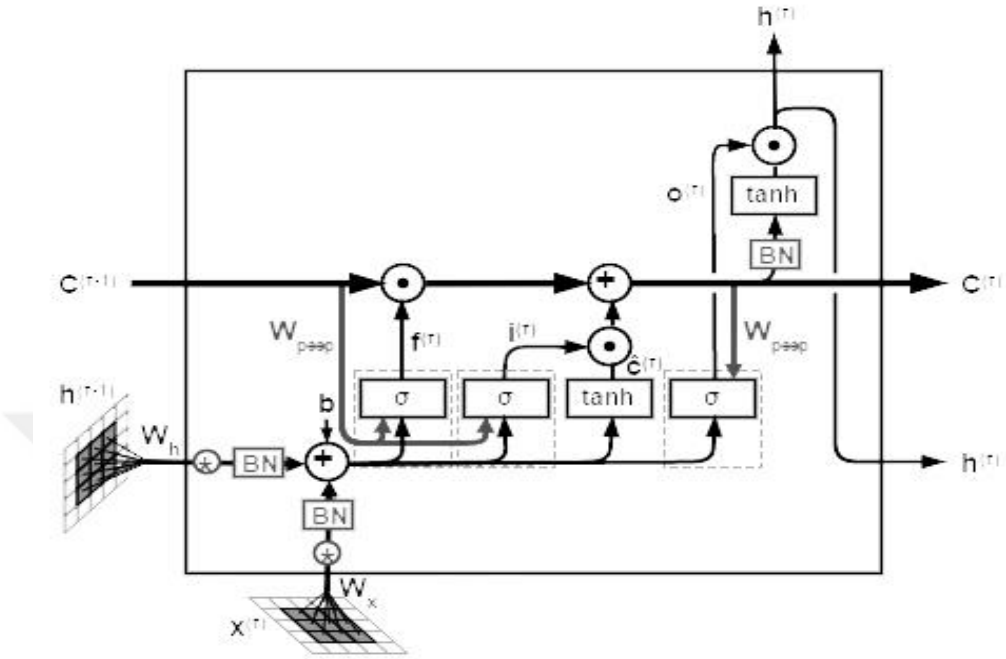


Şekil 3.11. CNN mimarisi

3.1.5.6. ConvLSTM

CNN, verilerin gizli özelliklerini uzay-zamansal olarak araştırma ve sınıflama konusunda kullanılırken, LSTM ise tipik sıralı zaman serisi verilerinden kullanılır. Bu iki algoritmanın avantajlarını birleştiren ConvLSTM modeli önerilmektedir. (Chen, 2020). ConvLSTM algoritması ilk olarak 2015 yılında Shi ve arkadaşları tarafından yağış tahmini için kullanıldı.

ConvLSTM modeli ile LSTM modeli içerisindeki işlemler konveksiyonel işlemlerle değiştirilmişlerdir. Bu değişimle algoritmanın klasik LSTM algoritmasına göre uzay-zamansal korelasyonları daha iyi yakaladığı gözlemlenmiştir. ConvLSTM, radar görüntüleri gibi boyut kavramının olduğu ve sıralamanında önemli olduğu verilerde de en iyi sonucu vermektedir. ConvLSTM temel şeması Şekil 3.12 ile gösterilmiştir.



Şekil 3.12. ConvLSTM yapısı.[6]

3.1.6. Yapay Zekâ Kütüphaneleri

Birçok yapay zekâ kütüphanesi ve paket programı bulunmaktadır. Bu çalışmada; Python programlama dili ile Keras ve TensorFlow 2.0 kütüphaneleri kullanılmıştır.

TensorFlow; Google tarafından 2015 yılında, uçtan uca açık kaynaklı bir makine öğrenme platformu olarak tasarlanan bir Python ve C++ ile kodlanmış bir derin öğrenme kütüphanedir. TensorFlow kararlı C++ ve Python API desteğinin yanında; Java, C#, Javascript ve R gibi dillerde de kullanılabilmektedir. CPU (Central Processing Unit) ve GPU (Graphics Processing Unit) desteği bulunmaktadır (Abadi, 2016).

Keras; TensorFlow üzerinde çalışan, Python ile kodlanmış, açık kaynak bir derin öğrenme kütüphanesidir. Keras ile yapay zekâ modellerinin kolayca

[6] https://medium.com/neuronio/an-introduction-to-convlstm-55c9025563a7

oluşturulabileceği, kullanıcı dostu bir programlama ortamı sağlamaktadır (Chollet, 2015).

3.1.7. Günlük Toplam Yağış Verisi

Meteorolojide yağış ölçüm birimi milimetre olup, bir metrekare alana kaç kilogram yağış olduğunu temsil eder. Yağış miktarı insanlı manuel ölçüm yapılan meteoroloji istasyonlarında plüviyometre ya da plüviyograf ile ölçülürken, insansız otomatik meteoroloji istasyonlarında (OMGİ) ise yağışölçer cihazı ile ölçüm yapılır.

Meteoroloji radarları doğrudan yağış miktarını ölçmezler. Bulut içerisinde ya da havadaki su damlacıklarını ölçerler. Yağış verisine ampirik Z ve R bağıntısı ile ulaşılır. Yağış verilerinin alınmasının nedeni yağışlı günlerin tespit edilmesi ve eğitim veri seti oluştururken bu yağışlı günlerin radar verilerinin alınması için kullanılmıştır.

3.2. Method

3.2.1. Verilerin Temin Edilmesi

Bu çalışmada kullanılan meteorolojik veriler; Hatay Radar verisi, Ek 8'de belirtilen istasyonlar için Günlük Toplam Yağış verisi MGM'nden alınmıştır. Ek 9'da Günlük toplam yağış verisi talep yazısı ile kullanım izin yazısı bulunurken, Ek 9'da Hatay radar verisi talep ve kullanım izin yazısı bulunmaktadır.

3.2.2. Verilerinin Hazırlanması

Yapay zekâ kullanılarak yapılan çalışmalarda, kullanılacak model ve algoritma kadar veri ön işleme aşaması önemli bir eşiktir. Veri, modelleme yapılmadan önce veri hazırlama aşamasından geçirilmektedir. Verinin hazırlanmasında ya da düzeltilmesinde kullanılan yöntemler, model sonucunda çıkan birçok parametreye etki etmektedir. Doğru veri hazırlanma süreci ve yöntemleri, modelleme hatalarını azaltır, sinir ağının eğitim sürecini hızlandırır, sistemin bir bütün olarak basitleşmesine ve daha doğru sonuçlara ulaşmasını

sağlamaktadır (Yu, 2005).

3.2.2.1. Radar Verilerinin Alınacağı günlerin tespiti

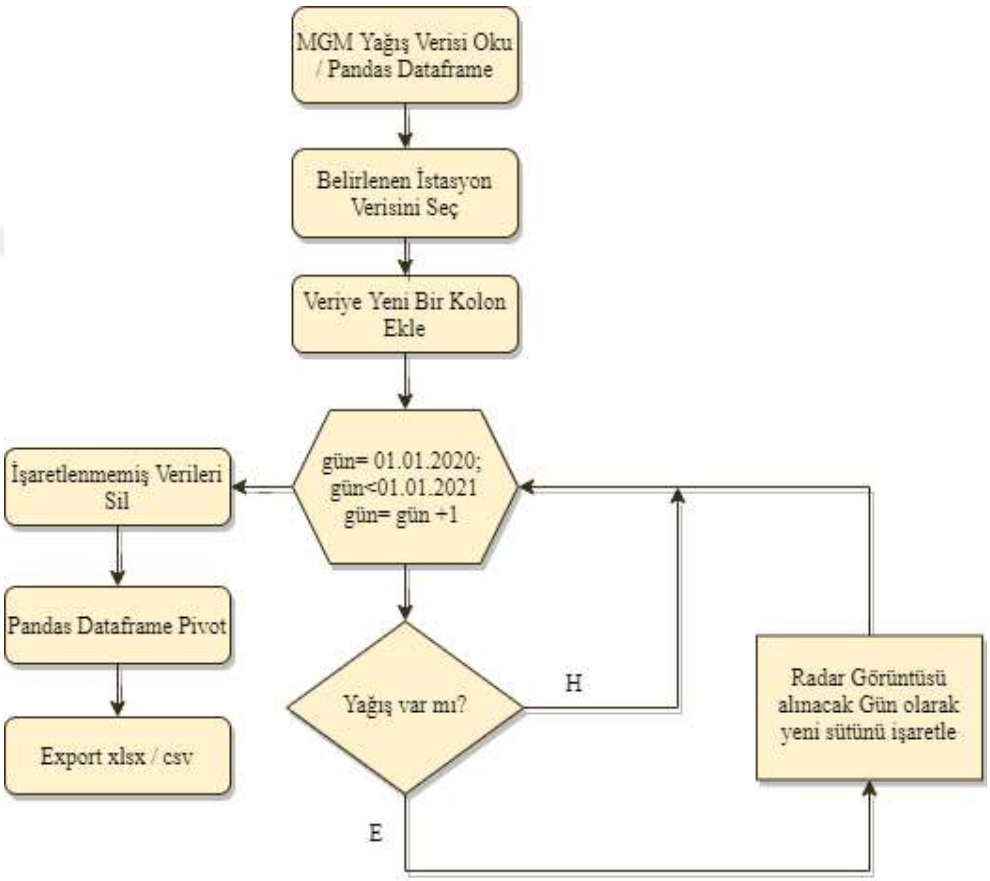
MGM radar ağında, 24 saat veri üretilmektedir. Fakat yılın her günü yağış olamayacağından dolayısı ile radar verisinde eko görülmeyeceğinden, havanın yağışlı olduğu günlerin toplam veri içerisinden ayıklanması gerekmektedir. Bunun en önemli nedeni veri sayısını düşürmek ve yapay sinir ağının ezbere kaçmasını önlemektir. Bu amaçla yağış verisi üzerinde çalışılmıştır. Görüntünün alınacağı bölge için seçilen istasyonların toplam yağış miktarı verisine bakılmıştır. Yağış verisi Omgi ve manuel ölçüm olarak iki ayrı veri olduğundan, seçilen istasyonlarda hangi tür ölçüm yapılıyorsa veri dosyalarından bulunan veri alınmıştır. Belirlenen alan için yağış miktarı var ise ilgili gün oluşturulan yağışlı günler tablosunda işaretlenmiştir. Bu aşamadan sonra işaretli günler için radar görüntüleri alınmıştır. Bu çalışma için bölge olarak Adana ve Mersin seçilip kullanılacak istasyonlar bu alandan seçilmiştir. Bu işlemde seçilen istasyonlar Çizelge 3.2'de gösterilmiştir. Örneğin; 01.01.2020 ile 31.12.2020 tarihleri arasında 177 gün için yağış kaydı olduğu bulunmuş ve bu 177 gün verisi alınmıştır.

Çizelge 3.3. Yağış verisi için seçilecek istasyonların listesi.

İstasyon Kodu	İstasyon Adı
17350	İncirlik Meydan Met. Md.
17351	Adana Meteoroloji Bölge Md.
17352	Adana Meydan Met. Md.
17340	Mersin Meteoroloji Md.
17320	Anamur Meteoroloji Md.
17355	Osmaniye Meteoroloji Md.
17330	Silifke Meteoroloji Md.
17978	Tarsus Meteoroloji Md.

Radar yağışlı günlerin belirlenmesi için MGM'nden Toplam Yağış Miktarı

Verisi üzerinde yapılan işlemler için akış şeması Şekil 3.13'te verilmiş olan Python programlama dilinde kodlanmış ve kullanılmıştır. İlgili Python betiği Ek 1'de verilmiştir.



Şekil 3.13. Yağışlı günlerin seçimi yapılırken kullanılan akış diagramı.

3.2.2.2. Radar Görüntü Özellikleri

Radar ham veri formatı çoğunlukla radar donanım ve yazılımını üreten firmanın tercihindedir. MGM Hatay radar verileri Vaisala Sigmet Iris formatında sunulmaktadır. IRIS kelimesi, Vaisala Sigmet Etkileşimli Radar Bilgi Sistemi (Vaisala Sigmet Interactive Radar Information System) kısaltmasıdır.

Radar verisi binary olarak kaydedilmiştir. Wradlib kütüphanesi ile ya da başka kütüphanelerde okunduğunda Dict veri tipinin bir alt sınıfı olan OrderDict formatında veri elde edilmektedir. Bu veri 'product_hdr', 'product_type', 'nbins', 'data' anahtarlarına sahiptir. 'product_hdr' görüntü ile ilgili temel bilgileri içermektedir. 'product_type' görüntünün tipini göstermektedir. 'data' ise görüntü ile ilgili piksel değerlerini içermektedir. Örnek olarak 1 Nisan 2020 tarihine ait PPI görüntüsünün başlık içeriği Ek 7'de verilmiştir.

Hatay radar PPI radar ürünü, 370km yarıçaplı kapsama alanına sahiptir. Kapsama alanı 720x720 adet piksel veri içermektedir. Her bir pikselin x ve y eksenleri boyunca uzunlukları eşit ve 984 m (~0,101 km) 'dir. Bir pikselin temsil ettiği alan yaklaşık 1 km2'dir.

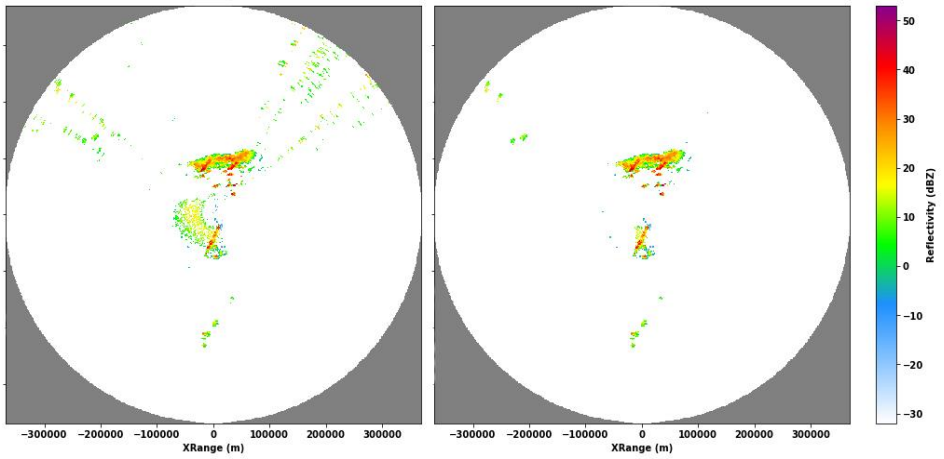
Başlangıçta 720x720 piksel boyutunda olan radar PPI görüntüsü Adana Çukurova bölgesini içine alacak şekilde 100x100 boyutunda alan kesilmiştir. Bu alan üzerinden model çalıştırılmıştır. 720x720 piksel olarak modele girdi olarak vermek yüksek donanıma sahip bir bilgisayar ve GPU desteği gerektirdiğinden bu işlem gerçekleştirilmiştir. Modelin uygulanabilirliği küçük bir alan üzerinden gösterilmiştir. Kesilen 100x100 alan şekil 3.14'te harita üzerinde gösterilmiştir.

Şekil 3.14. 100x100 olarak kesilen alan.

3.2.2.3. Radar Veri Dönüşümü

MGM'nden Hatay Radar görüntüleri; sıkıştırılmış günlük ham veri dosyalarının içerisinden yağışlı günler önceden hazırlanan yağışlı günler tablosundan faydalanarak alınmışlardır. Bu işlem için Ek 2'de verilen Python betiği kodlanıp kullanılmıştır.

Bu görüntüler içerisinden verilen zaman aralığına denk gelen görüntü seçilip Wradlib Kütüphanesi aracılığı ile okunduktan sonra üzerindeki radar hataları yine Wradlib Kütüphanesi aracılığı ile temizlenmiştir. Şekil 3.15'te yağış kaynaklı olmayan ekolar barındıran görüntü ile bu hatalı ekolardan arındırılmış görüntü verilmiştir.

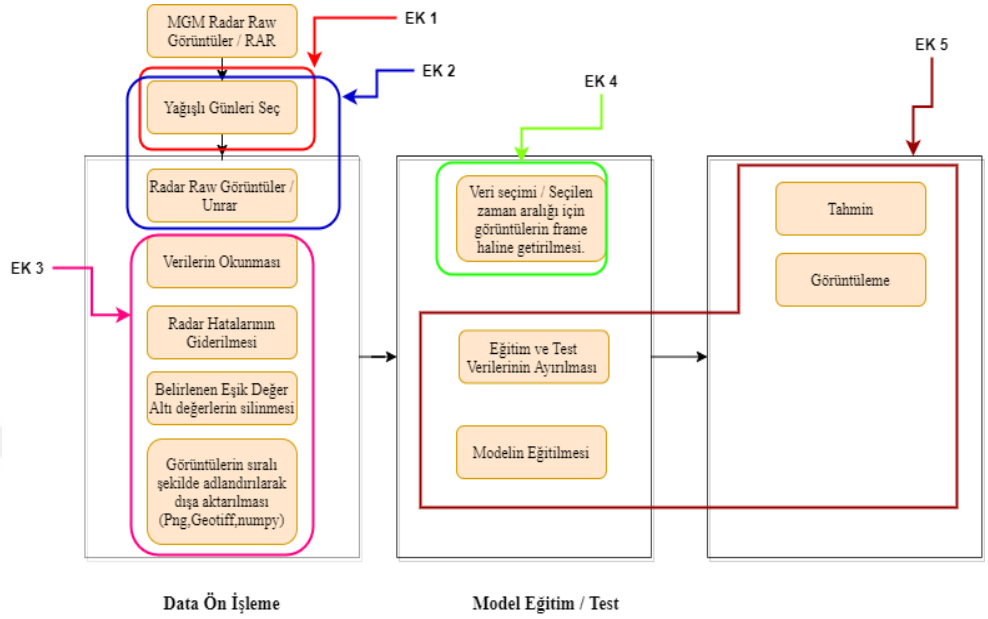


Şekil 3.15. Hatalı ekolar barındıran radar görüntüsü (solda) ve Wradlib kütüphanesi ise hatalı ekoların silindiği radar görüntüsü (sağda).

Minimum dBZ eşik değeri belirlenerek, bu değerin altındaki zayıf ekolar yağışı temsil etmeyen hatalı ekolar olarak kabul edilmişlerdir. Dolayısıyla 15dBZ 'ten küçük reflektivite değerler çalışmaya dâhil edilmemiştir (Mapiam, 2009).

Bu işlemden sonra okunan ve düzeltilen bu görüntüler oluşturulan klasörlere png, geotif ve numpy formatında dışa aktarılmıştır. Yapılan bu işlemler için Ek 3'te bulunan Python programı kodlanıp kullanılmıştır. Yapılan işlemleri Şekil 3.13'de Data Ön İşleme kısmında gösterilmiştir.

Kullanılan yapay öğrenme modeli görüntüyü; bir birini takip eden 6 görüntü ve her görüntüyü matris olacak şekilde kabul ettiğinden eğitim seçilen veriler yeniden şekillendirilip tek bir dosya halinde numpy array formatında kaydedilmişlerdir. Bu işlem için Ek 4'te bulunan Python betiği kodlanıp kullanılmıştır. Yapılan işlemleri işlemler Şekil 3.16'da Model Eğitim / Test kısmında gösterilmiştir.



Şekil 3.16. Data akış ve model diagramı.

3.2.3. ConvLSTM Modeli

Bu tez çalışmasında ConvLSTM modeli Python 3.9 versiyonu ile Keras Kütüphanesi kodlanmıştır.

ConvLSTM modeli oluşturmak için, görüntü girdilerini (batch_size, num_frames, width, height, channels) kabul edecek ve aynı yapıya sahip bir tahmin görüntüyü döndürecek 3 adet ConvLSTM2D katmanı kullanılmıştır. Bu katmanlar CNN yapısında eğitim aşamasında kaybolabilecek olan gradyana direnç sağlanması için katmanlardan sonra batch_normalization işlemi uygulanmıştır. Batch normalization CNN yapısını daha düzenli hale getirmek için kullanılan bir yöntemdir. Son olarak uzay-zamansal çıktı için Conv3D katmanı eklenmiştir.

Kullanılan görüntülerde frame sayısı / görüntü sayısı 6 olup, görüntü sadece eko değeri barındırdığından band sayısı da 1'dir. Görüntüler arasındaki zaman farkı 30 dakikadır. Hatay meteoroloji radarı 6 dakikada bir görüntü oluşturmaktadır. Fakat görüntünün hareketinin yakalanması için zaman aralığı 30 dakika seçilmiştir.

Bu çalışmada kullanılan optimizasyon fonksiyonu olarak Adam optimizasyonu kullanılmıştır. Adam optimizasyonu ağırlıklıları tekrarlama yolu güncellemek için kullanılan bir algoritmadır. Uygulaması basit olduğundan ve gürültülü durağan olmayan verilerin eğitilmesinde başarılı olduğundan kullanılmıştır.

Batch makine öğrenmesinde kullanılan eğitim örneklerinin sayısını ifade eder. YSA her seferinde batch size kadar veriyi aynı anda öğrenir. Batch size 2'nin katları şeklinde seçilebilir. Büyük batch size, bellek yetersizliğine neden olacağından batch size belirlenmeden önce kullanılan işlemci önbelleği miktarına bakılmalıdır. Bu çalışmada batch size 2 olarak belirlenmiştir.

Model epoch sayısı model eğitilirken verinin model tarafından kaç defa öğreneceğini kontrol eden parametredir. Bu değerin küçük olması eğitim süresini kısaltır fakat model tam öğrenememiş olabilir. Sayısının büyük olması ise eğitim süresini uzatır fakat model bu sayıya kadar eğitimi tamamlamış da olabilir. Yani gereksiz yere modeli eğitmeye devam etmiş olabiliriz. Uygun epoch; modelden modele daha çok probleme ve veri setine bağlı olarak değişmektedir. En doğru epoch sayısı belirleme yöntemi modelin gelişimini yani öğrenmeyi tamamladığı noktada eğitimi durdurmaktır. Bu çalışmada epoch sayısı 100 olarak verilmiş. Uygulamanın gelişimini tamamladığı noktada eğitimi durdurması için gerekli direktifler koda eklenmiştir. Python Keras kütüphanesi ile oluşturulan model ve tahmin kodları Ek 5'te bulunmaktadır. Model özeti çizelge 3.4'te verilmiştir.

Çizelge 3.4. Kullanılan ConvLSTM model özeti

Model: "model_1"		
Layer (type)	Output Shape	Param
input_1 (InputLayer)	(None, None, 100, 100, 1)	0
conv_lst_m2d_1 (ConvLSTM2D)	(None, None, 100, 100, 64)	416256
batch_normalization_1	(Batch (None, None, 100, 100, 64))	256
conv_lst_m2d_2 (ConvLSTM2D)	(None, None, 100, 100, 64)	295168
batch_normalization_2	(Batch (None, None, 100, 100, 64))	256
conv_lst_m2d_3 (ConvLSTM2D)	(None, None, 100, 100, 64)	33024
conv3d_1 (Conv3D)	(None, None, 100, 100, 1)	1729
Total params: 746,689		
Trainable params: 746,433		
Non-trainable params: 256		

Modele veri girdi olarak verilmeden önce ölçeklendirme yapılmıştır. Ölçekleme yöntemi olarak minimum maksimum ölçeklendirme yöntemi kullanılmıştır. Bu yöntem ile verinin yeni değeri; veri setindeki her veriden o veri setinde yer alan minimum değerden farkı alınarak ve veri setindeki maksimum ve minimum değerlerinin farkına oranlanarak hesaplanmaktadır. Bu veriyi normalize etmenin matematiksel formülü aşağıda verilmiştir.

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Veri seti eğitim verisi ve test verisi olarak ikiye ayrılmıştır. Bunun için veri seti %80 eğitim, %20 doğrulama veri seti olarak ayırılmıştır.

Kullanılan model yüksek bilgisayar donanımı ve özellikle GPU desteği gerektirdiğinden model Google Colaboratory (Colab) üzerinde çalıştırılmıştır. Colab

ile ücretsiz olarak bulut servisi üzerinden ve GPU desteği ile Keras, TensorFlow kütüphaneleri ile derin öğrenme uygulamaları geliştirilebilmektedir. Colab; Tesla P100 desteği sunmaktadır. Model belirtilen kartın GPU desteği ile eğitilmiştir.

4. BULGULAR ve TARTIŞMA

4.1. PPI Radar Görüntüleri

PPI ürünü atmosferi tek bir açıdan taradığı için yağışlı sistemin konumu ve şiddeti ile ilgili bir öngörü ürünüdür. Sadece PPI görüntüsüne bakılarak kuvvetli bir yağış uyarısı yapılması doğru değildir. Hacimsel taramalardan elde edilen görüntülerle desteklenmesi gerekmektedir. Bu tez çalışması ile yapılan işlemler ve sonucunda çıkan görüntü tahmini PPI üzerinden yapılsa da aynı model çok bandlı olarak diğer görüntüler için de yapılabilir.

4.2. Model Parametreleri

Yapay öğrenme modellerinde; veri setine ve probleme göre değişen algoritma ve yöntemlere modeli tasarlayanın karar vermesi gerekmektedir. Aynı zamanda modeli tasarlayan bu seçtiği tekniklere göre bazı parametreleri belirlemek durumundadır. Bu paramatre ya da hiperparametreler kısaca öğrenme sürecini kontrol etmek için girilen değişkenlerdir.

Radar görüntülerinin tahmini için Çizege 4.1'de verilen parametrelerle eğitim sağlanmıştır.

Çizelge 4.1. Model eğitim parametreleri

Eğitim veri seti	392 x 6
Doğrulama veri seti	49 x 6
Test veri seti	49 x 7
Optimizer	Adam
Öğrenme adım büyüklüğü /Öğrenme hızı	0.001
Kayıp fonksiyonu	Binary_crostopy
Veri paket boyutu (Batch Size)	2
Epoch Sayısı	100

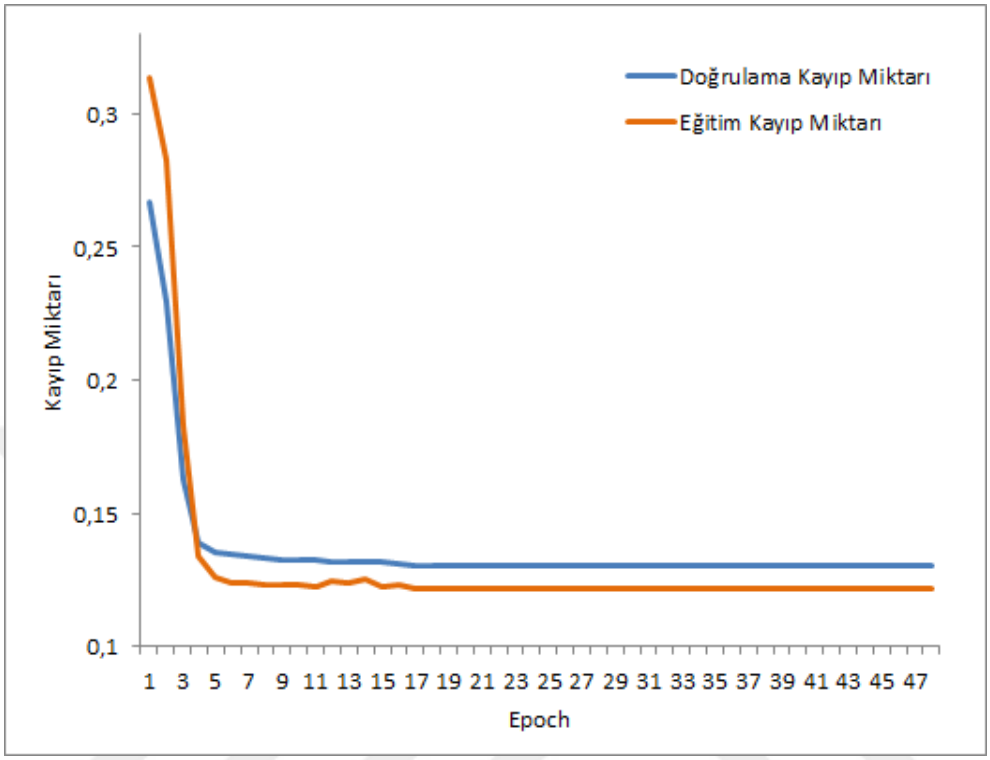
2989 adet görüntünün %80 eğitim, %10'u doğrulama ve %10'u test veri seti

olarak kullanılmıştır. Bu görüntüler sıralı halde, eğitim ve doğrulama veri setleri için frame olarak 6'şarlı gruplar haline getirilmişlerdir. Toplam 392 adet eğitim verisi bulunurken, 49 adet doğrulama verisi ve 49 adet test verisi bulunmaktadır. Model eğitildikten sonra test veri seti üzerinden tahmin yaptırılmıştır.

Epoch sayısı 100 olarak girilmiş fakat model eğitimde bir gelişme gösteremediğinde, eğitimin sonlanması için Keras Callbacks api sınıfları kullanılmıştır. Callbacks EarlyStopping sınıfı ile doğrulama veri setinin kayıp oranı takip edilmiş ve bu kayıp oranı 10 epoch boyunca bir gelişme gösteremediğinde ise eğitimin sonlandırılması sağlanmıştır. Aynı zamanda doğrulma kayıp miktarının bir gelişme gösteremediği durumda öğrenme adım sayısının azaltılması sağlanmış bu işlem de 5 epoch boyunca takip edilmesi için Callbacks ReduceLROnPlateau sınıfından faydalanılmıştır. Bu yöntemle modelin aşırı öğrenmesi engellenmiş ve model gelişimin durması ile eğitime son verilmesinden dolayı eğitim süresi kısaltılmıştır. Model eğitimi her çalışmada yaklaşık olarak 45-50 epoch arası eğitimi tamamladığı gözlenmiştir. Veriler arasındaki benzerlik azaldıkça yani veriler farklılaştıkça bu sayının artması beklenir.

4.3. Model Bulguları

Model kayıp miktarı ne kadar küçükse, modelin o kadar iyi öğrendiği anlamı taşımaktadır. Model kayıp miktarları, eğitim ve doğrulama veri setleri üzerinden hesaplanmaktadır. Modelin bu iki küme için ne kadar iyi performans gösterdiğinin rakamsal ifadesidir. Doğruluk oranının aksine, kayıp miktarı bir yüzde değildir. Eğitim veya doğrulama setlerinde her epoch için yapılan hataların toplamıdır. Model kayıp miktarları Şekil 4.1'de gösterilmiştir. Turuncu eğri eğitim veri seti kayıp miktarı iken, mavi çizgi ise doğrulama veri seti kayıp miktarıdır. Şekilde görüleceği üzere model 100 epoch ile eğitilse bile eğitim gelişim gösteremediğinden 47. epoch sürmüştür. Kayıp miktar eğrilerinin düşmesi modelin daha iyi ağırlık değerlerini yakaladığı anlamı taşırken, yükselmesi ise modelin aşırı eğitildiğinin göstergesidir.



Şekil 4.1. Model kayıp miktarları.

Model kayıp miktarlarının çok hızlı bir şekilde düşmesi ve bir süre sonra değişimin çok az ya da hiç olmaması model veri setinin küçük ya da homojen olmasından kaynaklanmaktadır. Model doğruluk oranı %80 -84 aralığında olduğu gözlenmiştir.

Model eğitildikten sonra, model tahmini ile gerçek görüntünün benzerlik oranları Kök Ortalama Kare Hatası (RMSE), Tepe Sinyali Gürültü Oranı (Peak Signal to Noise Ratio, PSNR) ve Yapısal Benzerlik Endeksi (Structural Similarity, SSIM) indexi ile ölçülmüştür. Bu ölçüm işlemi Python programlama dilinde kodlanan image-similarity-measures[7] paketi ile yapılmıştır.

RMSE, işleme nedeniyle piksel başına değişiklik miktarını ölçmektedir. RMSE değeri 0'dan ∞'a kadar değişebilmektedir. 0'a yakın negatif, yani düşük

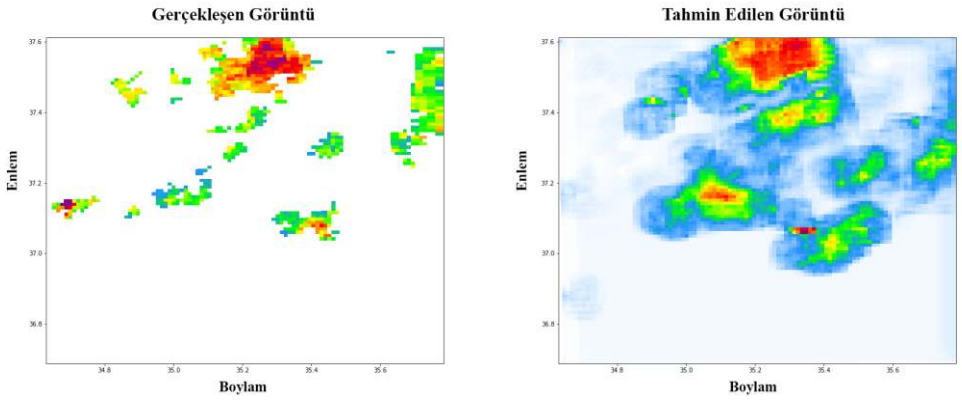
[7] https://github.com/up42/image-similarity-measures (Müller, 2020).

değerler iki görüntü arasında benzerlik oranının yüksekliğini göstermektedir. Değerin 0 olması iki görüntü ya da videonun aynı olduğunu ifade eder, değer ne kadar büyük ise görüntülerin benzerlik oranları o kadar az olmaktadır.

PSNR, referans görüntüyü yeniden oluşturulmuş görüntü ile karşılaştırır ve sayısal karşılaştırma yapar, ancak SSIM, insan görmesinin biyolojik faktörlerini hesaba katar ve aralarındaki yapısal benzerliği hesaplar (Dey, 2018). Karşılaştırılan iki görüntü aynı ise SSIM değeri 1 olmaktadır.

Model aralıklar eğitilmiş ve tahmin görüntüleri oluşturulmuştur. Bu tahminlerden elde edilen tahmin görüntüsü ile gerçekleşen görüntü arasındaki benzerlik oranları kaydedilmiştir. Elde edilen sonuçlara göre en iyi performansa sahip modelin tahminlerinin RMSE, PSNR ve SSIM değerleri sırasıyla 0.000013, 97.795, 0.998'dir.

Model Çizelge 4.4'te verilen parametreler ile çalıştırıldığında, ortaya çıkan tahmin sonuçları ve gerçekleşen radar görüntüleri karşılaştırmalı olarak Şekil 4.2, 4.3 ve 4.4'te verilmiştir. Şekillerde yatay eksen boylamı gösterirken dikey eksen ise enlemi göstermektedir. İlk görüntü radar tarafından alınan yani gerçekleşen görüntü, ikinci görüntü ise model tarafından tahmin edilen görüntüdür.



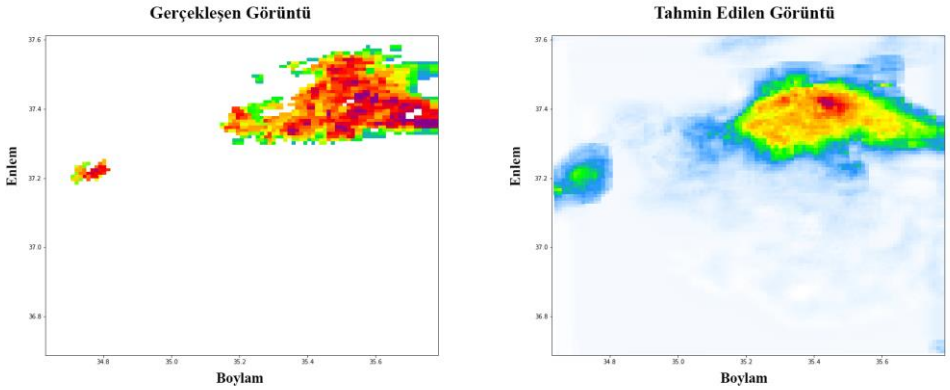
Şekil 4.2. Model çıktısı (a) Gerçekleşen görüntü (solda), tahmin edilen görüntü (Sağda)

Şekil 4.2'de görüldüğü üzere model tahmini, gerçek görüntü ile benzerlik

gösterse bile sistemin yoğunluk noktasının tahmin görüntüsünde kaydığı gözlenmiştir. İki görüntü arasındaki benzerlik oranları Çizelge 4.2'de gösterilmiştir.

Çizelge 4.2. Şekil 4.2 model çıktısının ve gerçekleşen görüntünün benzerlik oranları

RMSE	PSNR	SSIM
0.000039	88.714	0.981

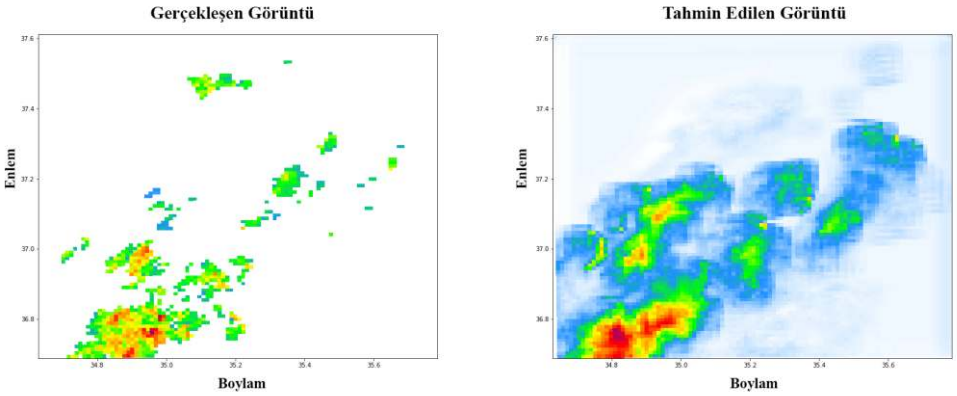


Şekil 4.3. Model çıktısı (b). Gerçekleşen görüntü (solda), tahmin edilen görüntü (Sağda)

Şekil 4.3'de Tahmin tutarlığı düşük gözlenmiş ve eko merkezinin taşınmadığı görülmektedir. İki görüntü arasındaki benzerlik oranları Çizelge 4.3'de gösterilmiştir.

Çizelge 4.3. Şekil 4.3 model çıktısının ve gerçekleşen görüntünün benzerlik oranları

RMSE	PSNR	SSIM
0.000042	87.534	0.989



Şekil 4.4. Model çıktısı (c) Gerçekleşen görüntü (solda), tahmin edilen görüntü (Sağda)

İki görüntü arasındaki benzerlik oranları Çizelge 4.4'de gösterilmiştir.

Çizelge 4.4. Şekil 4.4 model çıktısının ve gerçekleşen görüntünün benzerlik oranları

RMSE	PSNR	SSIM
0.000029	90.3	0.99

Model tahmin verisinde eko hareket yönünde ve etrafında mavi renkle gösterilen düşük ekolu bölgeler ya da izler bulunmaktadır. Bunun nedeni ConvLSTM algoritmasında kullanılan kapı aktivasyon fonksiyonlarıdır.(Elsayed, 2019). Bu aktivasyon fonksiyonlarının değiştirilmesi eko kümesinin izlerini tamamen silemediği gözlenmiştir.

Modele girdi olarak verilen 6 görüntü ve tahmin sonucu ile gerçekleşen görüntüler Ek 6'da ayrıntılı olarak verilmiştir.

Kullanılan convLSTM görüntülerinin zaman aralığı 30 dakika olarak seçilmiştir. Bu tercih ile modelin son görüntüden sonraki yarım saat için oluşabiliecek görüntüsünün tahmini amaçlanmıştır. Fakat daha uzun zaman aralıkları, aynı veriseti ile tahmin edilmek istenirse; model tahmin ettiği her görüntüyü var olan görüntüler dizisine ekleyerek ve diziden son altı görüntüyü giriş değeri kabul ederek tahmin işlemine devam edebilmesi sağlanarak sonuçlar

gözlenmiştir. Bu tahminlerde görüntülerin 2 ya da 3. görüntüden sonra bozulduğu ve gerçek görüntülerle bir benzerlik taşımadığı görülmüştür. Bu nedenle modelin seçilen zaman aralığından daha uzun vade de tahmin yapabilmesi için YSA eğitim veri setinin zaman aralığı ile ve tahmin edilmek istenen zaman aralığının eşit olması tahmin tutarlılığını yükselttiği gözlemlenmiştir.

5. SONUÇ ve ÖNERİLER

Bu tez çalışmasında; Hatay meteoroloji radarı PPI görüntüleri kullanılarak, ConvLSTM modeli ile YSA eğitilmiş ve henüz kayıt altına alınmamış görüntü ve görüntüler tahmin edilmeye çalışılmıştır. Radar PPI görüntüsü 370 km yarıçaplı bir alanı temsil etmektedir. 720x720 piksel alandan, Çukurova bölgesi 100x100 piksel bir alan kesilerek tahmin edilmiştir. Kullanılan ConvLSTM modelinin yüksek bilgisayar donanımı ve özellikle GPU desteği gerektirmesinden dolayı bu işlem gerçekleştirilmiştir. Model Google Colab üzerinden alınan üyelikle çalıştırılmıştır.

Tahmin yapılan alanın trasmsal faaliyet açısından yoğun bir bölge olması nedeni ile model sonuçlarının önemi artmaktadır. YSA ile yapılan eğitim sonucunda gerçekleşen görüntü ile tahmin edilen görüntü arasında %80-84 tutarlılık elde edilmiştir. Tutarlılık oranının artması model parametrelerine ve veri seti büyüklüğü ve çeşitliliğinebağlıdır. Bu tür görüntü problemlerinde veri seti büyüklüğü binler, onbinler ile yapılmalıdır. Bu açıdan kullanılan veri seti uzunluğu daha doğru tahmin ve yüksek oranlar için artırılmalıdır.

Eğitilen model test verisi üzerinden test edilmiştir. Test verileri modelin daha önce hiç karşılamadığı veriler olduklarından sonuçları önemlidir. Model tahmin sonucunda bir adet tahmin görüntüsü elde edilmiştir. Bu tahmin görüntüsü son görüntüden yarım saat sonra kaydedilecek görüntüdür. Tahmin edilen görüntü ile gerçekleşmiş görüntü görüntü karşılaştırma metrikleri ile karşılaştırılmıştır. Bu çalışmada kullanılan görüntü benzerlik metrikleri RMSE, PSNR ve SSIM'dir. Elde edilen sonuçlara göre, benzerlik metriklerinden RMSE 0.000088 - 0.000013 aralığında, PSNR 81.114 - 97.795 aralığında ve SSIM ise 0.95 – 0.998 aralığında ölçülmüştür. En iyi performansa sahip modelin tahminlerinin RMSE, PSNR ve SSIM değerleri sırasıyla 0.000013, 97.795, 0.998'dir.

Bu tez çalışmasından kullanılan model parametreleri, özellikle conv2d katmanlarındaki, konvilasyon işlemleri parametreleri değiştirilerek eğitilmiştir. Yapılan tahminler ile gerçek görüntü arasındaki benzerlik oranları her seferinde

kaydedilmiştir. Bu benzerlik oranları; RMSE benzerlik değerinin 0.000088 ile 0.000013 arasında

ConvLSTM modeli çok kanallı olarak çalıştırılabilmektedir. Bu çalışmada tek band üzerinden işlem yapılmıştır. Fakat eklenen her band bir bilgisayar performans maliyeti gerektirse de, tahmin sonuçlarını iyileştirebilmektedir. Çok kanallı model için aşağıdaki veriler, aynı zamanı temsil edecek şekilde oluşturulan veri setine eklenebilir.

- CAPPI ya da MAX gibi hacimsel veri içeren ürünler.
- Radar rüzgâr verileri.
- Yıldırım Tespit ve Takip Sistemi (YTTS) verileri.
- Sayısal Yükseklik Modeli verisi

Bu tez çalışması için gerekli olan görüntülerin sıralı olması gerekmektedir. Radar bakımları ve zaman zaman donanımsal sıkıntılar dolayısı ile düzenli bir veri akışı sağlanamamaktadır. Örneğin bu çalışmada 30 dakikalık görüntüler ile çalışıldığından, MGM'nden alınan tüm radar görüntüleri arasından sadece saat başı ve yarımda görüntü alınmaktadır. 6 görüntü ile çalışıldığından 3 saatlik veri akışının kesintisiz olması gerekmektedir. Eksik veriler alınamayacağından, veri setinin genişletilmesi zorlaşmaktadır. ConvLSTM gibi görüntü bazlı sıralı veri isteyen algoritmalar için veri setinin büyük olması ve çeşitlendirilmesi modelin daha tutarlı sonuçlar üretebilmesini sağlamaktadır.

Meteoroloji radar verileri formatlarından, projeksiyonlarından ve radar donanım ya da yazılım sağlayıcıları tarafından farklı bir formatta kaydedildiklerinden okunması zor bir veridir. Bu tez çalışması ile radar verisinin okunması, işlenmesi ve veri format değişiklikleri yapılmıştır. Yapılan işlemler eklerde verilmiştir. Bu işlemler aynı ya da benzer çalışmalar için birer örnek teşkil edeceği gibi kolaylaştırıcı adım ya da ön ayak olacağı düşünülmektedir.

KAYNAKLAR

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X., 2016, Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation (OSDI),n. 16, p. 265-283.

Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C. and Hickey, J., 2019. Machine learning for precipitation nowcasting from radar images. arXiv preprint arXiv:1912.12132.

Ahmadi N., Constandinou T. & Bouganis C., 2019. Decoding Hand Kinematics from Local Field Potentials Using Long Short-Term Memory (LSTM) Network. 9th International IEEE EMBS Conference on Neural Engineering (NER 2019), 1-5s.

Ahrens, C. D., & Henson, R. 2016. Meteorology today, an introduction to weather, climate and the environment . Boston: Cengage Learning, 11th ed., 662 pp.

Albawi, S., Mohammed, T. A., & Al-Zawi, S., 2017. Understanding of a convolutional neural network. International Conference on Engineering and Technology (ICET), IEEE, pp. 1-6.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z., 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In International conference on machine learning, PMLR, pp. 173-182.

Apaydın, E., 2012, Yapay Öğrenme, Boğaziçi Üniversitesi Yayınevi, İstanbul,496s

Atlas, D., 1990. Editor. Radar in Meteorology: Battan Memorial and 40th Anniversary Radar Meteorology Conference. American Meteorological Society, Boston, MA, USA.

Bahdanau, D., Cho, K., & Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bakır, A. K., 2018. Kalman Filtreleme Yaklaşımı Kullanılarak Yağışölçer Verisi ile

Meteoroloji Radarı Yağış Tahmininin İyileştirilmesi. Master's thesis, İstanbul Üniversitesi- Cerrahpaşa.

Chen, X., Xie, X., & Teng, D., 2020. Short-term Traffic Flow Prediction Based on ConvLSTM Model. In 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), IEEE, pp. 846-850.

Chollet, F., & others. (2015). Keras. https://keras.io. Erişim tarihi: 17.06.2021

Chumchean, S., Sharma, A., & Seed, A. (2003). Radar rainfall error variance and its impact on radar rainfall calibration. Physics and Chemistry of the Earth, Parts A/B/C, 28(1-3), 27-39.

Dey, N., Ashour, A. S., Shi, F., & Balas, V. E., 2018, Soft Computing Based Medical Image Analysis. Academic Press, Chapter 5, Pages 61-79.

DiPietro, R., & Hager, G. D., 2020. Deep learning: RNNs and LSTM. In Handbook of medical image computing and computer assisted intervention. Academic Press, pp. 503-519.

Doviak, R. J., Zrnic, D. S., & Schotland, R. M., 1994. Doppler radar and weather observations. Applied Optics, 33(21), 4531.

Elsayed, N., Maida, A., Bayoumi, M. et al., 2019, Effects of different activation functions for unsupervised convolutional lstm spatiotemporal learning. Advances in Science, Technology and Engineering Systems Journal, 4(10.25046).

Fabry, F., 2015. Radar Meteorology Principles and Practice. Cambridge University Press, ISBN : 978-1-107-07046-2.

Fang, W., Jiang, J., Lu, S., Gong, Y., Tao, Y., Tang, Y., ... & Liu, J., 2020, A LSTM algorithm estimating pseudo measurements for aiding INS during GNSS signal outages, Remote sensing, 12(2), 256.

Fang, X., Shao, A., Yue, X., Liu, W. et al., 2018., Statistics of the Z–R relationship for strong convective weather over the Yangtze–Huaihe River basin and its application to radar reflectivity data assimilation for a heavy rain event. Journal of Meteorological Research, 32(4), 598-611.

Fente, D. N., & Singh, D. K., 2018. Weather forecasting using artificial neural network. In 2018 second international conference on inventive communication and computational technologies (ICICCT), IEEE, pp. 1757-1761.

Fukao, S., Hamazu, K., 2014. Radar for Meteorological and Atmospheric Observations. Springer Japan, ISBN: 978-4-431-54333-6.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... & Ng, A. Y. ,2014. Deep speech: Scaling up end-to-end speech recognition. arXiv preprint arXiv:1412.5567.

Hasan, M.T., Fattahul Islam, K.M., Rahman, M.S. & Li, S., 2019. Weather Forecasting Using Artificial Neural Network. In: Sun, X., Pan, Z. Bertino E. (eds) Artificial Intelligence and Security. ICAIS 2019. Lecture Notes in Computer Science, Springer, vol 11633. https://doi.org/10.1007/978-3-030-24265-7_15

Heistermann, M., Jacobi, S., & Pfaff, T., 2013. An open source library for processing weather radar data (wradlib). Hydrology and Earth System Sciences, 17(2), 863-871.

Helmus, J.J. & Collis, S.M., 2016. The Python ARM Radar Toolkit (Py-ART), a Library for Working with Weather Radar Data in the Python Programming Language. Journal of Open Research Software. 4(1), p.e25. Doi: http://doi.org/10.5334/jors.119

Heye, A., Venkatesan, K., & Cain, J., 2017. Precipitation nowcasting: Leveraging deep recurrent convolutional neural networks. Proceedings of the Cray User Group (CUG).

Hochreiter, S., & Schmidhuber, J., 1997. Long short-term memory. Neural computation, 9(8), 1735-1780.

Hochreiter, S., 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 6(02), 107-116.

Huffman GJ, Bolvin DT, Nelkin EJ., 2015. Integrated Multi-satellitE Retrievals for GPM (IMERG) technical documentation. NASA/GSFC Code. 2015;612:47.

IEEE Standard Association, 2020. IEEE Standard Letter Designations for Radar-Frequency Bands. in IEEE Std 521-2019 (Revision of IEEE Std 521-2002) pp.1-15.

Jiang, T., Gradus, J. L., & Rosellini, A. J., 2020. Supervised machine learning: a brief primer. Behavior Therapy, 51(5), 675-687.

Joselin, J., Dinesh, T., & Ashiq, M., 2018. A review on neural networks. Int. J. Trend Sci. Res. Dev.(IJTSRD), 2, 565-569.

Kaelbling, L. P., Littman, M. L., & Moore, A. W., 1996. Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285.

Kim, S., Hong, S., Joh, M., & Song, S. K., 2017. DeepRain: ConvLSTM Network for Precipitation Prediction using Multichannel Radar Data. arXiv e-prints, arXiv-1711.

Koval, S. I., 2018. Data preparation for neural network data analysis, IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow and St. Petersburg, Russia, pp. 898-90.

Kumar, A., Islam, T., Sekimoto, Y., Mattmann, C., & Wilson, B., 2020. Convcast: An embedded convolutional LSTM based architecture for precipitation nowcasting using satellite data. Plos one, 15(3), e0230114.

LeCun, Y., Bengio, Y., & Hinton, G., 2015. Deep learning. nature, 521(7553), 436-444.

Li, N., Wang, Z., Sun, K., Chu, Z., Leng, L., & Lv, X., 2017. A quality control method of ground-based weather radar data based on statistics. IEEE Transactions on Geoscience and Remote Sensing, 56(4), 2211-2219.

Lin, T. Y., Winner, K., Bernstein, G., Mittal, A., Dokter, A. M., Horton, K. G., ... & Sheldon, D., 2019. MistNet: Measuring historical bird migration in the US using archived weather radar data and convolutional neural networks. Methods in Ecology and Evolution, 10(11), 1908-1922.

Mapiam, P. P., Sriwongsitanon, N., Chumchean, S., & Sharma, A. 2009. Effects of rain gauge temporal resolution on the specification of a Z–R relationship. Journal of Atmospheric and Oceanic Technology, 26(7), 1302-1314.

McCarthy, N., Guyot, A., Dowdy, A., & McGowan, H., 2019. Wildfire and weather radar: A review. Journal of Geophysical Research: Atmospheres, 124(1), 266-286.

Mesnard, F., & Sauvageot, H., 2010. Climatology of anomalous propagation radar echoes in a coastal area. Journal of Applied Meteorology and Climatology, 49(11), 2285-2300.

Meteoroloji Genel Müdürlüğü, 2021. Radar Meteorolojisi: Meteoroloji Genel Müdürlüğü Radar Şebekesi. https://www.mgm.gov.tr/genel/meteorolojiradarlari.aspx?s=radaragi Erişim Tarihi: 28.06.2021

Müller, M. U., Ekhtiari, N., Almeida, R. M., & Rieke, C., 2020, Super-resolution of multispectral satellite images using convolutional neural networks. arXiv preprint arXiv:2002.00580.

Nilsson, C., Dokter, A. M., Verlinden, L., Shamoun-Baranes, J., Schmid, B., Desmet, P., ... & Liechti, F., 2019. Revealing patterns of nocturnal migration using the European weather radar network. Ecography, 42(5), 876-886.

Niu, D., Diao, L., Xu, L., Zang, Z., Chen, X., & Liang, S., 2020. Precipitation Forecast Based on Multi-channel ConvLSTM and 3D-CNN. In 2020 International Conference on Unmanned Aircraft Systems (ICUAS) , IEEE, pp. 367-371.

Roodschild, M., Sardiñas, J. G., & Will, A., 2020, A new approach for the vanishing gradient problem on sigmoid activation. Progress in Artificial Intelligence, 9(4), 351-360.

Schmidt, J., Marques, M.R.G., Botti, S. et al., 2019. Recent advances and applications of machine learning in solid-state materials science. npj

Comput Mater 5, 83.

Shi, X., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C., 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. arXiv preprint arXiv:1506.04214.

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. ,2017. Deep learning for precipitation nowcasting: A benchmark and a new model. arXiv preprint arXiv:1706.03458.

Sokol, Z., Szturc, J., Orellana-Alvear, J., Popová, J., Jurczyk, A., & Célleri, R., 2021. The role of weather radar in rainfall estimation and its application in meteorological and hydrological modelling—A Review. Remote Sensing, 13(3), 351.

Sun, J., Xue, M., Wilson, J. W., Zawadzki, I., Ballard, S. P., Onvlee-Hooimeyer, J., ... & Pinto, J., 2014. Use of NWP for nowcasting convective precipitation: Recent progress and challenges. Bulletin of the American Meteorological Society, 95(3), 409-426.

Thorndahl, S., Einfalt, T., Willems, P., Nielsen, J. E., Veldhuis, M. C. T., Arnbjerg-Nielsen, K., ... & Molnar, P., 2017. Weather radar rainfall data in urban hydrology. Hydrology and Earth System Sciences, 21(3), 1359-1380.

Villarini G, Krajewski W.F., 2009. Review of the Different Sources of Uncertainty in Single Polarization Radar-Based Estimates of Rainfall, Surv Geophys (2010), 31:107- 129.

WMO, 2006. Instruments And Observıng Methods Report No. 88. https://library.wmo.int/doc_num.php?explnum_id=9317 Erişim tarihi: 15.06.2021.

WMO, 2014. Guide to Meteorological Instruments and Methods of Observations Part I –Chapter 7: Measurement of Precipitation. WMO-No.8, 2014 Edition (updated 2017), ISBN : 978-92-63-10008-5.

WMO, 2017. Guidelines for Nowcasting Techniques. WMO-No.1198, 2017 Edition, ISBN:978-92-63-10008-5

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J., 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

Xu, J., Ren, L., Fan, H., Mao, E., & Liu, Q., 2018. Clutter and range ambiguity suppression using diverse pulse train in pulse Doppler system. Sensors, 18(7), 2326.

Yu, L., Wang, S., & Lai, K. K., 2005. An integrated data preparation scheme for neural network data analysis. IEEE Transactions on Knowledge and Data Engineering, 18(2), 217-230.

Zhang, C., Wang, H., Zeng, J., Ma, L., & Guan, L., 2020. Tiny-RainNet: A Deep CNN-BiLSTM Model for Short-Term Rainfall Prediction. Geophysical Research Letters.

Zhang, L., Huang, Z., Liu, W., Guo, Z., & Zhang, Z., 2021, Weather radar echo prediction method based on convolution neural network and Long Short-Term memory networks for sustainable e-agriculture. Journal of Cleaner Production, 298, 126776.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., & Liu, J., 2017. LSTM network: a deep learning approach for short-term traffic forecast. IET Intelligent Transport Systems, 11(2), 68-75.

ÖZGEÇMİŞ

1985 doğumlu olup, orta öğretimini Ankara'da Anadolu Meteoroloji Meslek lisesinden 2004 yılında mezun olarak tamamladı. 2005 yılında başladığı Süleyman Demirel Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Sistemleri Öğretmenliğini 2010 yılında bitirdi. 2014 yılında ise Hoca Ahmet Yesevi Üniversitesi Bilgisayar Mühendisliği bölümünden mezun oldu. 2019 yılında Çukurova Üniversitesi Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilimdalı'nda tezli yüksek lisans yapmaya başladı. 2020 yılında mühendislik tamamlama ile girdiği Mersin Üniversitesi Bilgisayar Mühendisliği bölümüne devam etmektedir.

2004 yılından itibaren Meteoroloji Genel Müdürlüğü'nde, Bölgesel Tahmin ve Erken Uyarı Merkezi ve havalimanlarında istidlalci olarak çalışmaktadır.

EKLER

Ek 1. Toplam Yağış Miktarı Verisinden Yağışlı Günlerin Tespiti.

```
 # -*- coding: utf-8 -*-
import numpy as np
import pandas as pd
import os
# yağışlı günleri al
def get_rainy_days(data=None,ist_no=None,is_save=False,yil=2020):
    df = data[data["YIL"]>=yil]
    df = df[df['Istasyon_No'] == ist_no]
    if(is_save):
        fname = '.'.join([str(ist_no),'xlsx'])
        df.to_excel(fname,index=None)
        fname = '.'.join([str(ist_no),'txt'])
        df.to_csv(fname,sep=";",index=None)
    return df
# bütün istasyonlar için yağışlı günleri kaydet
def pivot_tables_all(data=None,yil=2020):
    data = data[data["YIL"]>=yil]
    table = pd.pivot_table(data, values='TOPLAM_YAGIS_Manuel_Omgi_mm2',
index='Istasyon_No',columns='Tarih')
    table.to_excel("03-EDITED DATA\\03-
Pivot_tables_Yagis_Tum_Istasyonlar.xlsx")
    return table
#istenilen alan ya da bölge için yağışlı günleri al kaydet
def pivot_tables_obs(data=None,ists=None,filename='04-Istenilen_istasyonlar'):
    data = data[data['Istasyon_No'].isin(ists)]
    fname = os.path.join('03-EDITED DATA',filename+ ".xlsx")
    table = pd.pivot_table(data, values='TOPLAM_YAGIS_Manuel_Omgi_mm2',
index=['Istasyon_No','Istasyon_Adi'],columns='Tarih')
```

```
    table.to_excel(fname)
    return table

#Mgm data oku
filepath_manuel = "01-MGM DATA//20210104551A-Günlük Toplam Yağış
MANUEL.txt"
filepath_omgi = "01-MGM DATA//20210104551A-Günlük Toplam Yağış
OMGI.txt"
filepath_readme = "01-MGM DATA//20210104551A_Beni_Oku!!!.txt"
filepath_secili_ist = "01-MGM DATA//secili_istasyonlar.txt"
dfm = pd.read_csv(filepath_manuel,sep="|")
dfo = pd.read_csv(filepath_omgi,sep="|")
dfr = pd.read_csv(filepath_readme,sep="|")
dfc = pd.read_csv(filepath_secili_ist,sep="|")

dfo.loc[dfo.TOPLAM_YAGIS_mm == 0.0,'TOPLAM_YAGIS_mm'] = np.nan
manuel = dfm.Istasyon_No.unique()
omgi = dfo.Istasyon_No.unique()
print(len(omgi))
print(len(manuel))
# omgi ya da manuel olanları al
newlist = []
for om in omgi:
    for man in manuel:
        if(om == man):
            newlist.append(man)
dfo = dfo[dfo.Istasyon_No.isin(newlist)==False]
cols = ['Istasyon_No', 'Istasyon_Adi', 'YIL', 'AY', 'GUN',
       'TOPLAM_YAGIS_Manuel_Omgi_mm']
```

```
# birleştir.
dfo.columns = cols
dfm.columns = cols
frames = [dfm,dfo]
dfson = pd.concat(frames)
dfson = dfson.sort_values(by=['Istasyon_No','YIL', 'AY', 'GUN'])
lst = dfson.TOPLAM_YAGIS_Manuel_Omgi_mm.values.copy()
dfson["TOPLAM_YAGIS_Manuel_Omgi_mm2"] = lst
dfson = dfson.drop('TOPLAM_YAGIS_Manuel_Omgi_mm', 1)
dfson = dfson.dropna()
dfson.rename(columns={"TOPLAM_YAGIS_Manuel_Omgi_mm2":
"TOPLAM_YAGIS_Manuel_Omgi_mm"})
cols = ['Istasyon_No', 'Istasyon_Adi','Tarih','GUN','AY',
'YIL','TOPLAM_YAGIS_Manuel_Omgi_mm2']

dfson["Tarih"] = dfson["AY"].astype(str) + '.' + dfson["GUN"].astype(str)+ '.' +
dfson["YIL"].astype(str)
dfson = dfson[cols]
dfson["Tarih"] = pd.to_datetime(dfson['Tarih'])
#dfson["Tarih"] = pd.to_datetime(dfson['Tarih'],format="%m/%d/%Y")
dfson = dfson[dfson["YIL"]>=2020]
dfson = dfson[dfson["YIL"]<2021]

dfson.loc[dfson.TOPLAM_YAGIS_Manuel_Omgi_mm2 == -
1.0,'TOPLAM_YAGIS_Manuel_Omgi_mm2'] = np.nan

dfson.to_excel("03-EDITED DATA\\01-Yagisli_Gunler.xlsx",index=None)
istss = dfc.Istasyon_Numarasi.unique().tolist()
dfis = pivot_tables_obs(data=dfson,ists=istss)
```

```
dfall = pivot_tables_all(data=dfson)
len(dfall.columns)

print(len(dfis.columns))

#yagisli gğnlerin tarihlerini kaydet
yagisliGunler = dfis.columns.tolist()
dfyg = pd.DataFrame(yagisliGunler)
dfyg.columns = ["tarih"]
dfyg.to_excel("03-EDITED DATA\\05-Yagisli Gun Tarihleri.xlsx")

dfyg.to_csv("03-EDITED DATA\\06-Yagisli Gun Tarihleri.txt")
```

Yağışlı Günlere Ait Radar Görüntülerinin Seçilmesi

```
# -*- coding: utf-8 -*-
"""
@author: alişan aktay
- rar dosyalarını unrar eder.
"""
import pandas as pd
import numpy as np
import glob
import os
import tarfile
import warnings

warnings.filterwarnings('ignore')

#rar dosyasını alır o dosyanın ait olduğu günde
#yağış var ise unrar eder
def unrar_files():
    filelist = glob.glob("00-2019 RAR FILES/*.gz")
    destinationFolder = "02-2019 UNRAR FILES"
    file_number = 0
    print("toplam dosya sayısı : " , len(filelist))
    for item in filelist:
        pre, ext = os.path.splitext(os.path.basename(item))
        date = pre.split('.')
        date = date[3]
        dest = os.path.join(destinationFolder, pre[:-4])
        print(str(file_number +1) + " / ", len(filelist) )
        print(item)
```

```
        if not os.path.exists(dest):
            os.makedirs(dest)

        # open file
        file = tarfile.open(item)
        # extracting file
        file.extractall(dest)
        file.close()
        file_number += 1

#dosya adını yaz
def write_file_name(r_name=None):
    print("\n")
    cols = ["radar","year","month","day","hour","minute"]
    a = [[x,y] for x,y in zip(cols,r_name)]
    for i in range(0,len(a)-1):
        print(a[i][0] + "  ------>   " + a[i][1])
#dosya adını yaz
def extract_filename(fname=None):
    pre, ext = os.path.splitext(os.path.basename(fname))
    radar = pre[:3]
    year = pre[3:5]
    month =pre[5:7]
    day = pre[7:9]
    hour = pre[9:11]
    minute = pre[11:13]
    return [radar,year,month,day,hour,minute]
if __name__ == "__main__":
    unrar_files()
```

Ek 3. Radar Görüntülerinin Okunması / Clutter Temizleme ve Dışa Aktarılması

```
# -*- coding: utf-8 -*-
"""
Created on Sun Jun 27 20:21:46 2021

@author: varto
"""

import wradlib as wrl
import matplotlib as mpl
import matplotlib.pyplot as pl
import warnings
import numpy as np
warnings.filterwarnings('ignore')
import glob
import os
import h5py
import random
from numpy import random
import pprint
from sklearn.preprocessing import MinMaxScaler

try:
    get_ipython().magic("matplotlib inline")
except:
    pl.ion()
import matplotlib.colors as col
```

```
def check_folder_create_if_not(directory=None):
    if not os.path.exists(directory):
        os.makedirs(directory)

#raw data folder
raw_folder = './02-2020 UNRAR FILES'
sub_raw_folders = [name for name in os.listdir(raw_folder) if
os.path.isdir(os.path.join(raw_folder, name))]

#numpy array kayıt klasoru
npy_save_folder = 'HTY_NUMPY_ARRAY_2020'
check_folder_create_if_not(npy_save_folder)

#radar goruntu png kayıt klasoru
png_save_folder = 'HTY_PNG_2020'
check_folder_create_if_not(png_save_folder)

#radar geotiff kayıt dosyası
geotiff_save_folder = 'HTY_GEOTIFF_2020'
check_folder_create_if_not(geotiff_save_folder)

# verilen dosya adından parametreleri ayırır
def extract_filename(fname=None):
    pre, ext = os.path.splitext(os.path.basename(fname))
    print(pre)
    #HTY 20 12 09 00 18 03
    radar = pre[:3]
    year = pre[3:5]
```

```
    month =pre[5:7]
    day = pre[7:9]
    hour = pre[9:11]
    minute = pre[11:13]
    return [radar,year,month,day,hour,minute]

def time_interval(minutes=None):
    if(minutes==30):
        return ["00","30"]
    elif(minutes==45):
        return ["00","15","30","45"]
    elif(minutes==60):
        return ["00"]

# plot data
def plot_data(data_=None,xy_=None,title=None,fig_size=(10,8),dpi_res=72):
    fig = pl.figure(figsize=fig_size,dpi=dpi_res)
    ax = fig.add_subplot(111)
    pm = ax.pcolormesh(xy_[..., 0], xy_[..., 1],
                       np.ma.masked_equal(data_, -9999.),
                        cmap=cmap)
    pl.colorbar(pm, label='Reflectivity (dBZ)')
    ax.set_xlabel('Longitude')
    ax.set_ylabel('Latitude')
    pl.show()

def get_wradlib_cmap():
    startcolor = 'white'
    color1= '#8ec7ff'
```

```
    color2 = 'dodgerblue'
    color3 = 'lime'
    color4 = 'yellow'
    color5 = 'darkorange'
    color6 = 'red'
    endcolor = 'darkmagenta'
    colors = [startcolor, color1, color2, color3, color4, color5, color6, endcolor]
    return col.LinearSegmentedColormap.from_list('wrl1',colors)

def export_raw_to_geotiff(data_=None,xy_=None,proj_=None,fname=None):
    #data_[data_ <=0.] = -9999.
    fname = fname + '.tif'
    geotif_file_name = os.path.join(geotiff_save_folder, fname)
    ds = wrl.georef.create_raster_dataset(data_, xy_, projection=proj_)
    wrl.io.write_raster_dataset(geotif_file_name, ds, 'GTiff')
    del ds

def
export_raw_to_geotiff_window(data_=None,xy_=None,proj_=None,fname=None)
:
    data_[data_ <=0.] = -9999.
    data_[data_ >=0.] = -9999.
    data_[:,0:5] = 255
    data_[0:5,:] = 255
    data_[714:719,:] = 255
    data_[:,714:719] = 255
    pl.imshow(data_)
    fname = fname + '.tif'
    geotif_file_name = os.path.join(geotiff_save_folder, fname)
```

```
    ds = wrl.georef.create_raster_dataset(data_, xy_, projection=proj_)
    wrl.io.write_raster_dataset(geotif_file_name, ds, 'GTiff')
    del ds
# export png
def export_raw_to_image(data_=None,fname=None,dpi_=300,cmap_=None):
    fname = fname + '.png'
    png_file_name = os.path.join(png_save_folder, fname)
    mpl.image.imsave(png_file_name, data_,cmap=cmap,dpi=72)

def export_raw_to_numpy_array(data_=None,fname=None):
    fname = fname + '.npy'
    npy_file_name = os.path.join(npy_save_folder,fname)
    np.save(npy_file_name,data_)

def remove_small_dbz(ppi_data=None,small_then_15=True,nandata=0.0):
    ppi_data[ppi_data < 0.] = nandata
    if(small_then_15):
        ppi_data[ppi_data < 15.] = nandata
    return ppi_data

def remove_clutter(ppi_data=None,xy_=None,show_plot=False):
    clutter = wrl.clutter.filter_gabella(ppi_data, tr1=12, n_p=6, tr2=1.1)
    if(show_plot):
        plot_data(clutter,xy_,title="Clutter Map")
    data_no_clutter = wrl.ipol.interpolate_polar(ppi_data, clutter)
    if(show_plot):
        plot_data(data_no_clutter,xy_,title="Removed Culutters")
    return data_no_clutter
```

```
def export(image=False,gtif=False,npy=False,time_interval_=30):
    for folder in sub_raw_folders:
        filelist = glob.glob("02-2020 UNRAR FILES/"+folder+"/*.*")
        #random.shuffle(filelist)
        print(folder)
        for filename in filelist:
            radar,year,month,day,hour,minute = extract_filename(filename)
            minuteList = time_interval(time_interval_)
            if(minute in minuteList):
                #print(file)
                ds = wrl.io.open_raster(filename)
                #pprint.pprint(ds.GetMetadata())
                data, xy, proj = wrl.georef.extract_raster_dataset(ds, nodata=-9999.)
                proj_wgs84 = wrl.georef.epsg_to_osr(4326)
                xy1 = wrl.georef.reproject(xy, projection_source=proj,
projection_target=proj_wgs84)

                #data[data < 0.] = 0.
                data = remove_clutter(data,xy1)
                data = remove_small_dbz(data)

                #print(data.shape)
                #print(xy.shape)
                #print(proj)

                # bölgesel
                alandata = data[220:320,260:360].copy()
                alanxy = xy1[220:320,260:360].copy()
```

```
            pre, ext = os.path.splitext(os.path.basename(filename))
            pre = pre[:-2]
            print(filename)
            if(image):
                print("\t Exporting png....")
                export_raw_to_image(data_=alandata,fname=pre)
            if(npy):
                print("\t Exporting numpy array....")
                export_raw_to_numpy_array(alandata,pre)
            if(gtif):
                print("\t Exporting geoTiff....")

export_raw_to_geotiff(data_=alandata,xy_=alanxy,proj_=proj_wgs84,fname=pre)

#export_raw_to_geotiff_window(data_=alandata,xy_=alanxy,proj_=proj_wgs84,fn
ame=pre)



cmap = get_wradlib_cmap()
cmap.set_bad(color='grey')

filelist = glob.glob("02-2020 UNRAR FILES/radar.HTY.PPI.20200401/*.*")
num = random.randint(0,len(filelist))
print(num)
filename = filelist[num]

ds = wrl.io.open_raster(filename)
```

```
#pprint.pprint(ds.GetMetadata())
data, xy, proj = wrl.georef.extract_raster_dataset(ds, nodata=-9999.)
proj_wgs84 = wrl.georef.epsg_to_osr(4326)
xy1 = wrl.georef.reproject(xy, projection_source=proj,
projection_target=proj_wgs84)

print("\nPLOT RAW DATA\n")
#plot_data(data,xy1)
#data[data < 0.] = 0.
print("\nPLOT RAW DATA\n")
data = remove_clutter(data)
#plot_data(data,xy1)
data = remove_small_dbz(data)

# bölgesel
alandata = data[220:320,260:360]
alanxy = xy1[220:320,260:360]

#çiz
plot_data(alandata,alanxy)
pre, ext = os.path.splitext(os.path.basename(filename))
#export_raw_to_image(data=alandata,fname=pre)
print(num)

# EXPORT İŞLEMİ
export(image=True,gtif=True,npy=True)
```

Ek 4. Seçilen Görüntüler Üzerinde İşlemler / Verinin Hazırlanması

```
# -*- coding: utf-8 -*-
"""
@author: alişsan aktay
"""

import pandas as pd
import numpy as np
import glob
import os
import shutil
import matplotlib.pyplot as pl
from sklearn.preprocessing import MinMaxScaler

def check_folder_create_if_not(directory=None):
    if not os.path.exists(directory):
        os.makedirs(directory)




def int_to_str_zero(integer_value=None):
    return str(integer_value) if integer_value>=10 else '0'+str(integer_value)

# alt klasor oluştur
def
create_selected_folder_start_stop(domain_path=None,numstart=320,numstop=500
):
    for num in range(numstart,numstop+1):
        path = os.path.join(domain_path, int_to_str_zero(num))
```

```
        check_folder_create_if_not(path)

# alt klasor oluştur
def create_selected_folder(domain_path=None,_num=100):
    for num in range(1,_num+1):
        path = os.path.join(domain_path, int_to_str_zero(_num))
        check_folder_create_if_not(path)

# npy ve gtif alt klasorleri oluştur
def create_folder_from_dirList(dlist=None,main_folder=None):
    for _folder in dlist:
        _path = os.path.join(main_folder,_folder)
        check_folder_create_if_not(_path)

#png klasor içeriğini tarar npy gtif içerik olarak listeler
def
count_folder_files_png(domain_path_png=None,npy_path=None,gtif_path=None)
:
    dirListing = os.listdir(domain_path_png)
    # npy ve gtif klasor oluştur
    create_folder_from_dirList(dirListing,main_folder=HTY_SELECTED_NPY)
    create_folder_from_dirList(dirListing,main_folder=HTY_SELECTED_GTIF)

    _fileList_npy = []
    _fileList_gtif = []
    # npy ve gtif listelerini oluştur
    for _folder in dirListing:
        _png_path = os.path.join(domain_path_png, _folder)
        _npy_path = os.path.join(npy_path, _folder)
```

```
        _gtif_path = os.path.join(gtif_path, _folder)
        _flist_npy = [os.path.join(_npy_path, name) for name in os.listdir(_png_path)
if os.path.isfile(os.path.join(_png_path, name))]
        _flist_gtif = [os.path.join(_gtif_path, name) for name in os.listdir(_png_path)
if os.path.isfile(os.path.join(_png_path, name))]
        # kontrol
        if(len(_flist_npy)!=7):
            print("\n DIKKAT HATALI KLASOR")
            print(_folder)
        _fileList_npy.append(_flist_npy)
        _fileList_gtif.append(_flist_gtif)
    _fileListNPY = np.array(_fileList_npy,dtype=object)
    _fileListGTIF = np.array(_fileList_gtif,dtype=object)

    # npy yap
    for row in range(0,_fileListNPY.shape[0]):
        for col in range(0, _fileListNPY.shape[1]):
            _fileListNPY[row][col]  = _fileListNPY[row][col][:-3] + 'npy'
            _fileListGTIF[row][col]  = _fileListGTIF[row][col][:-3] + 'tif'

    return _fileListNPY,_fileListGTIF

# npy dosyaları kopyalar
def copy_npy_files(_filesList=None):
    os.chdir('..')
    _errFiles = []
    for row in range(0,_filesList.shape[0]):
        for col in range(0, _filesList.shape[1]):
            _baseFileName = os.path.basename(_filesList[row][col])
```

```
            _dFilePath = os.path.join(main_selected_folder, _filesList[row][col])
            _sFilePath = os.path.join(source_npy_folder, _baseFileName)
            print("kopyalanıyor....")
            print("\t",_sFilePath)
            try:
                shutil.copyfile(_sFilePath,_dFilePath)
            except:
                _errFiles.append([_sFilePath,_dFilePath])
                continue
    os.chdir(main_selected_folder)
    return _errFiles

 # geoTIFF dosyaları kopyalar
def copy_gtif_files(_filesList=None):
    os.chdir('..')
    _errFiles = []
    for row in range(0,_filesList.shape[0]):
        for col in range(0, _filesList.shape[1]):
            _baseFileName = os.path.basename(_filesList[row][col])
            _dFilePath = os.path.join(main_selected_folder, _filesList[row][col])
            _sFilePath = os.path.join(source_gtif_folder, _baseFileName)
            print("kopyalanıyor....")
            print("\t",_sFilePath)
            try:
                shutil.copyfile(_sFilePath,_dFilePath)
            except:
                _errFiles.append([_sFilePath,_dFilePath])
                continue
    os.chdir(main_selected_folder)
```

```
    return _errFiles

# frame oluştur
def create_frames(_filesList=None):
    _framesNpyAll = []
    for row in range(0,_filesList.shape[0]):
        _framesNpy = []
        for col in range(0, _filesList.shape[1]):
            _fpath = _filesList[row][col]
            _frameNpy = np.load(_fpath)
            _framesNpy.append(_frameNpy)
        _framesNpyAll.append(_framesNpy)
    return np.array(_framesNpyAll)

# gonderilen klasorde ve alt klasorlerdeki dosya listesini geri gonderiri
def get_file_names(domain_path=None):
    dirListing = os.listdir(domain_path)
    _fileList = []
    for _folder in dirListing:
        _path = os.path.join(domain_path, _folder)
        _fList = [os.path.join(_path, name) for name in os.listdir(_path) if
os.path.isfile(os.path.join(_path, name))]
        _fileList.append(_fList)
    return np.array(_fileList)

# normalize data
# min-max normalization
def normalize_data(data_=None):
    print(data_.shape)
```

```
    print(data_.min().round(5), data_.max().round(5)) # -1, 1
    scaler = MinMaxScaler()
    data_ = scaler.fit_transform(data_.reshape(data_.shape[0], -
1)).reshape(data_.shape)
    print(data_.shape)
    print(data_.min().round(5), data_.max().round(5)) # -1, 1
    return data_

if __name__ == "__main__":
    #selected gtif
    HTY_SELECTED_GTIF = "HTY_SELECTED_GTIF"
    check_folder_create_if_not(HTY_SELECTED_GTIF)

    #selected npy
    HTY_SELECTED_NPY = 'HTY_SELECTED_NPY'
    check_folder_create_if_not(HTY_SELECTED_NPY)

    #selected png
    HTY_SELECTED_PNG = 'HTY_SELECTED_PNG'
    check_folder_create_if_not(HTY_SELECTED_PNG)

    #source npy
    source_npy_folder = 'HTY_NUMPY_ARRAY_2020'

    #source npy
    source_gtif_folder = 'HTY_GEOTIFF_2020'

    #seçilen ana klasor
    main_selected_folder = 'HTY_SELECTED_2020'
```

```
    npy_list,gtif_ist =
count_folder_files_png(domain_path_png=HTY_SELECTED_PNG,npy_path=HT
Y_SELECTED_NPY,
                                    gtif_path=HTY_SELECTED_GTIF)
    frames = create_frames(npy_list)
    frames = normalize_data(frames)

    frames = np.swapaxes(frames, 0, 1)
    np.save("frames_100x100_30_dakika_2020.npy",frames)
```

Ek 5. ConvLSTM Modeli

```
# -*- coding: utf-8 -*-
"""
convLSTM Modeli
"""

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import io
from sklearn.preprocessing import MinMaxScaler
# import imageio
from IPython.display import Image, display
# from ipywidgets import widgets, Layout, HBox
import matplotlib.colors as col

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,
Activation,ConvLSTM2D,Flatten,Dropout

from mpl_toolkits.axes_grid1 import ImageGrid
import matplotlib.animation as animation
import imageio

import imageio
from IPython.display import Image, display
from ipywidgets import widgets, Layout, HBox
```

```
import shutil
import matplotlib.pyplot as pl
from sklearn.preprocessing import MinMaxScaler

import cv2
import image_similarity_measures
from image_similarity_measures.quality_metrics import rmse, psnr

import pandas as pd
import seaborn as sns

import cv2
import image_similarity_measures
from image_similarity_measures.quality_metrics import rmse, psnr,ssim

# renklendirme
def get_wradlib_cmap():
    startcolor = 'white'
    color1= '#8ec7ff'
    color2 = 'dodgerblue'
    color3 = 'lime'
    color4 = 'yellow'
    color5 = 'darkorange'
    color6 = 'red'
    color7 = 'pink'
    endcolor = 'darkmagenta'
    colors = [startcolor, color1, color2, color3, color4, color5, color6,
color7,endcolor]
    return col.LinearSegmentedColormap.from_list('wrl1',colors)
```

```
xy_ = np.load('drive/Tez_colab/alanxy_100x100_220-320_260-360.npy')
cmap = get_wradlib_cmap()
cmap.set_bad(color='grey')

def plot_frame_from_frames(d_set=None):
    cmap = get_wradlib_cmap()
    cmap.set_bad(color='grey')
    fig, axes = plt.subplots(3, 3, figsize=(10, 8))
    data_choice = np.random.choice(range(len(d_set)), size=1)[0]
    for idx, ax in enumerate(axes.flat):
        if(idx<7):
            a = d_set[data_choice][idx]
            a = a.reshape((100,100))
            ax.imshow(a, cmap=cmap)
            ax.set_title(f"Frame {idx + 1}")
            ax.axis("off")

    print(f"Seçilen örnek {data_choice}.")
    plt.show()

def plot_frame_from_frames_prediction(d_set=None):
    cmap = get_wradlib_cmap()
    cmap.set_bad(color='grey')
    fig, axes = plt.subplots(2, 4, figsize=(10, 8))
    for idx, ax in enumerate(axes.flat):
        if(idx<7):
            a = d_set[idx][idx]
            a = a.reshape((100,100))
            ax.imshow(a, cmap=cmap)
```

```
            ax.set_title(f"Frame {idx + 1}")
            ax.axis("off")

    plt.show()

# plot data
def
plot_data(data_=None,xy_=None,title=None,fig_size=(10,10),dpi_res=300,is_save
=False,legent=False):
    params = {'axes.labelsize': 16,
          'axes.titlesize': 16,
          'axes.facecolor': 'white'}
    plt.rcParams.update(params)
    fig = plt.figure(figsize=fig_size,dpi=dpi_res)
    ax = fig.add_subplot(111)
    pm = ax.pcolormesh(xy_[..., 0], xy_[..., 1],
                       np.ma.masked_equal(data_, -9999.),
                       cmap=cmap)
    if(legent):
      plt.colorbar(pm, label='Reflectivity (dBZ)')
    ax.set_title(title,fontname="Times New Roman",fontsize=14)
    ax.set_xlabel('Boylam',fontname="Times New Roman",fontsize=14)
    ax.set_ylabel('Enlem',fontname="Times New Roman",fontsize=14)

    plt.show()

    if(is_save):
      _fname= 'drive/Tez_colab/images_pr/' + title
      fig.savefig(fname=_fname, dpi=fig.dpi, facecolor='w', edgecolor='w',
```

```
        orientation='portrait', papertype=None, format=None,
        transparent=False, bbox_inches=None, pad_inches=0.3,
        frameon=None, metadata=None)

def plot_seven_img():
    #plt.imshow(new_prediction[5].reshape((100,100)))

#*****************************************************************
    new_one = np.concatenate((example,predicted_frame))
    print(example.shape)
    print(new_one.shape)

    cmap = get_wradlib_cmap()
    cmap.set_bad(color='grey')



    fig = plt.figure(figsize=(20., 20.),dpi=150)
    grid = ImageGrid(fig, 111,
                     nrows_ncols=(3, 3),
                     axes_pad=0.3,
                     )

    xy_ = np.load('drive/Tez_colab/alanxy_100x100_220-320_260-360.npy')
    cmap = get_wradlib_cmap()
    cmap.set_bad(color='grey')

    #pm = plt.pcolormesh(xy_[..., 0], xy_[...,
1],np.ma.masked_equal(im.reshape((100,100)), -9999.),cmap=cmap)
```

```
    #plt.colorbar(pm, label='Reflectivity (dBZ)')


    frame_no = 1
    for ax, im in zip(grid, new_one):
        # Iterating over the grid returns the Axes.
        if(frame_no==7):
            ax.set_title("Gerçekleşen Görüntü ")
        elif (frame_no==8):
            ax.set_title("Tahmin Görüntüsü")
        else:
            ax.set_title("Görüntü " + str(frame_no))
        ax.set_xticks([])
        ax.set_yticks([])
        img = im.reshape((100,100))
        ax.imshow(img,cmap=cmap,)
        frame_no += 1

    plt.show()
def load_prepare_data():
    # datayı al
    #dset = np.load("drive/Tez_colab/frames_100x100_30_dakika.npy")
    dset = np.load("drive/Tez_colab/frames_100x100_30_dakika_3.npy")
    dset = np.swapaxes(dset, 0, 1)
    # band ekle
    dset = np.expand_dims(dset, axis=-1)
    return dset

def create_shifted_frames(data):
```

```
    x = data[:, 0 : data.shape[1] - 1, :, :]
    y = data[:, 1 : data.shape[1], :, :]
    return x, y

def get_model_movingm():
  inputs = layers.Input(shape=(None, *x_train.shape[2:]))

  x = layers.ConvLSTM2D(
      filters=64,
      kernel_size=(5, 5),
      padding="same",
      return_sequences=True,
      activation="relu",
      name = "inputs"
  )(inputs)
  x = layers.BatchNormalization()(x)
  x = layers.ConvLSTM2D(
      filters=64,
      kernel_size=(3, 3),
      padding="same",
      return_sequences=True,
      activation="relu",
  )(x)
  x = layers.BatchNormalization()(x)
  x = layers.ConvLSTM2D(
      filters=64,
      kernel_size=(1, 1),
      padding="same",
      return_sequences=True,
```

```
    activation="relu",
  )(x)
  output = layers.Conv3D(
    filters=1, kernel_size=(3, 3, 3), activation="sigmoid", padding="same"
  )(x)

  # buil model
  model = keras.models.Model(inputs, output)
  model.compile(
    loss=keras.losses.binary_crossentropy, optimizer=keras.optimizers.Adam(),
    metrics=[keras.metrics.BinaryAccuracy()]
  )
  return model

dataset = load_prepare_data()

#eğitim ve val seç
indexes = np.arange(dataset.shape[0])
np.random.shuffle(indexes)
train_index = indexes[: int(0.8 * dataset.shape[0])]
val_index = indexes[int(0.8 * dataset.shape[0]) :]
train_dataset = dataset[train_index]
val_dataset = dataset[val_index]

# etiketleme
x_train, y_train = create_shifted_frames(train_dataset)
x_val, y_val = create_shifted_frames(val_dataset)

# data shapes
```

```
print("Training Dataset Shapes: " + str(x_train.shape) + ", " + str(y_train.shape))
print("Validation Dataset Shapes: " + str(x_val.shape) + ", " + str(y_val.shape))

print("Training Dataset Shapes Y : " + str(x_train.shape) + ", " + str(y_train.shape))
print("Validation Dataset Shapes Y : " + str(x_val.shape) + ", " + str(y_val.shape))

#modeli al
model = get_model_movingm()
model.summary()

# Modeli yaz ve kaydet
from keras.utils import plot_model
plot_model(model, to_file='drive/Tez_colab/model.png',
show_shapes=True,show_layer_names=True)

# Modeli Çiz
tf.keras.utils.plot_model(
    model,
    to_file="drive/Tez_colab/model.png",
    show_shapes=True,
    show_dtype=True,
    show_layer_names=True,
    rankdir="TB",
    expand_nested=False,
    dpi=96,
)

# Modeli eğit
```

```
# callbacks
radar_callbacks = [
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=10),
    keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5),
    keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-
{val_loss:.2f}.h5'),
    keras.callbacks.TensorBoard(log_dir='./logs'),
]

early_stopping = keras.callbacks.EarlyStopping(monitor="val_loss", patience=10)
reduce_lr = keras.callbacks.ReduceLROnPlateau(monitor="val_loss", patience=5)

# hiperparametreler.
epochs_num = 100
batch_size = 2

# model fit
history = model.fit(
    x_train,
    y_train,
    batch_size = batch_size,
    epochs=epochs_num,
    validation_data=(x_val, y_val),
    callbacks=radar_callbacks,
)

# ************************************************************
# Model sonuçlarını kaydet
model.save("drive/Tez_colab/save_model_s/my_model")
```

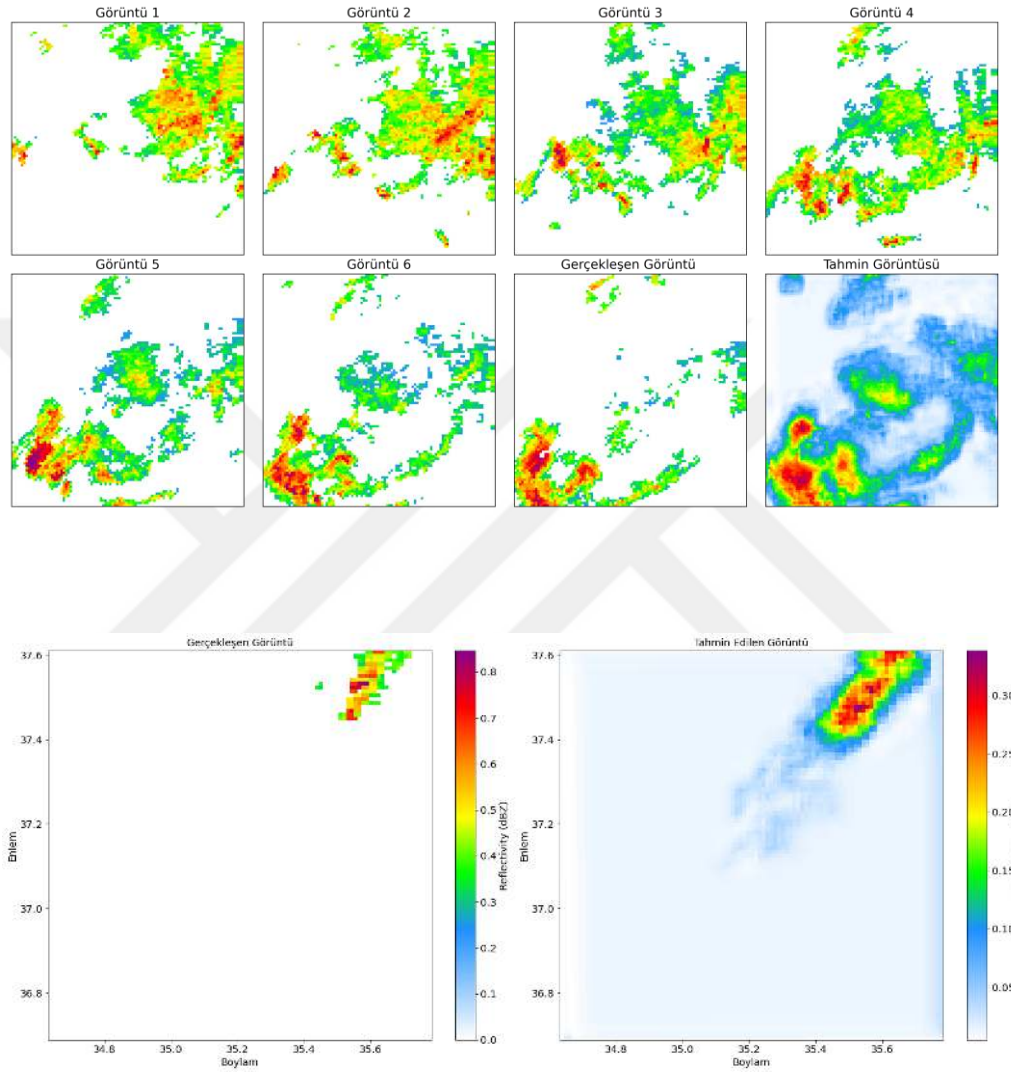
```
# ************************************************************
# loss eğirisi çiz
tloss = np.array(history.history['loss'])
vloss = np.array(history.history['val_loss'])

plt.figure(figsize=(10,10),dpi=72)
plt.plot(tloss)
plt.plot(vloss)
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch',)
plt.legend(['train_loss', 'val_loss'], loc='upper right')
plt.show()
plt.savefig("drive/Tez_colab/deneme.png")

# ************************************************************
# ^kayıp miktarlarını kaydet
print(tloss.shape)
print(vloss.shape)
np.save("drive/Tez_colab/vloss.npy",vloss)
np.save("drive/Tez_colab/tloss.npy",tloss)

# ************************************************************
example = val_dataset[np.random.choice(range(len(val_dataset)), size=1)[0]]
print(example.shape)
# Pick the first/last  frames from the example.
frames = example[:6, ...]
print(frames.shape)
```

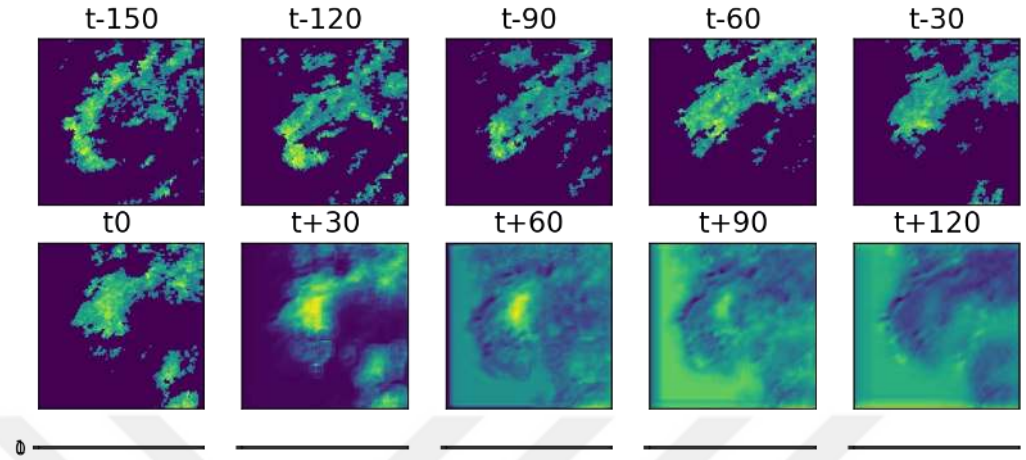
```
new_prediction = model.predict(np.expand_dims(frames, axis=0)) #input haline
getir
print(new_prediction.shape)
new_prediction = np.squeeze(new_prediction, axis=0)
print(new_prediction.shape)
predicted_frame = np.expand_dims(new_prediction[-1, ...], axis=0)
print(predicted_frame.shape)


#plt.imshow(new_prediction[5].reshape((100,100)))

new_one = np.concatenate((example,predicted_frame))
print(example.shape)
print(new_one.shape)

gercekselen = new_one[6,...].reshape((100,100))
tahmin_edilen = new_one[7,...].reshape((100,100))
plot_data(gercekselen,xy_=xy_,fig_size=(12,10),dpi_res=72,title="Gerçekleşen
Görüntü",legent=False)
plot_data(tahmin_edilen,xy_=xy_,fig_size=(12,10),dpi_res=72,title="Tahmin
Edilen Görüntü",legent=False)

truth_img = cv2.cvtColor(gercekselen, cv2.COLOR_GRAY2BGR)
predict_img = cv2.cvtColor(tahmin_edilen, cv2.COLOR_GRAY2BGR)

out_rmse = rmse(truth_img, predict_img)
out_psnr = psnr(truth_img, predict_img)
out_ssim = ssim(truth_img, predict_img)
```

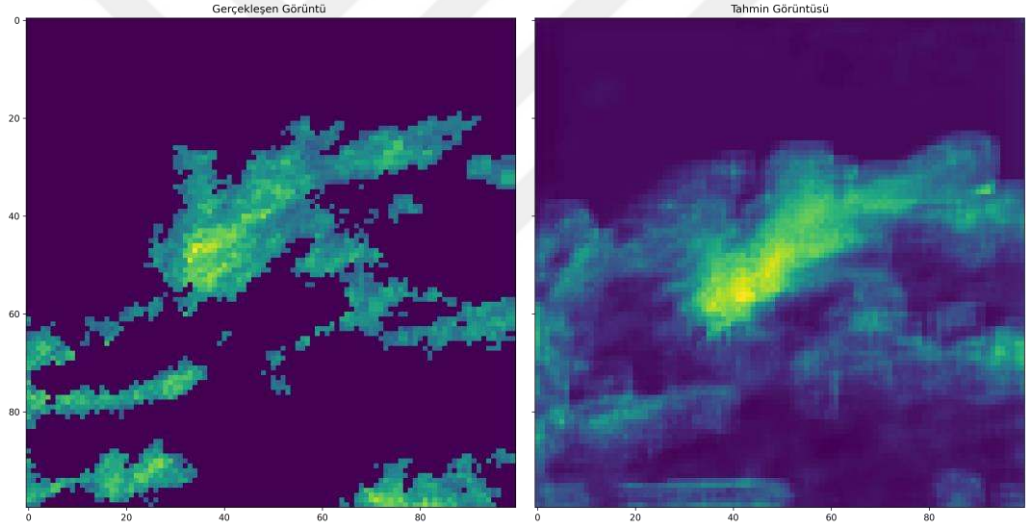
```
rmse_value = '{:f}'.format(out_rmse)

print("RMSE : ",rmse_value,"     PSNR : ",round(out_psnr,5), "     SSIM :
",round(out_ssim,5))

# ************************************************************
example = val_dataset[np.random.choice(range(len(val_dataset)), size=1)[0]]
# ilk 6 goruntuyu al
frames = example[:6, ...]

print(frames.shape)

#3x0.5 =1.5 saat
#3x0.5 =1.5 saat
kac_frame_sonrasi = 4

print("frames.shape   : ",frames.shape)

for fm in range(kac_frame_sonrasi):
  print("fm : ",fm)
  new_prediction = model.predict(np.expand_dims(frames[-6:], axis=0)) #input
haline getir
  new_prediction = np.squeeze(new_prediction, axis=0)
  print("new_prediction :", new_prediction.shape)
  predicted_frame = np.expand_dims(new_prediction[-1, ...], axis=0)
  print("predicted_frame.shape  : ", predicted_frame.shape)
  frames = np.concatenate((frames,predicted_frame))
  print("frames.shape   : ",frames.shape)
```

```
fig = plt.figure(figsize=(10., 10.),dpi=150)
grid = ImageGrid(fig, 111,
                 nrows_ncols=(3, 5),
                 axes_pad=0.3,
                 )

frame_no = 1
for ax, im in zip(grid, frames):
    # Iterating over the grid returns the Axes.
    if(frame_no<6):
        ax.set_title("t-" + str((6-frame_no)*30))
    elif (frame_no==6):
      ax.set_title("t0 ")
    else:
        ax.set_title("t+" + str((frame_no-6)*30) )

    ax.set_xticks([])
    ax.set_yticks([])
    img = im.reshape((100,100))
    ax.imshow(img)
    frame_no += 1

plt.show()
```

Ek 6. Tahmin Sonuçları Görselleri




t-150
t-120
t-90
t-60
t-30
t0
t+30
t+60
t+90
t+120


Gerçekleşen Görüntü
Tahmin Görüntüsü
0
20
40
60
80

Ek 7. 1 Nisan 2020 Tarihine Ait PPI Dosyası Başlık ve Kısaca İçeriği

```
OrderedDict([('product_hdr',
              OrderedDict([('structure_header',
                            OrderedDict([('structure_identifier', 27),
                                         ('format_version', 8),
                                         ('bytes_in_structure', 519040),
                                         ('flag', 0)])),
                           ('product_configuration',
                            OrderedDict([('structure_header',
                                          OrderedDict([('structure_identifier',
                                                        26),
                                                       ('format_version', 6),
                                                       ('bytes_in_structure',
                                                        320),
                                                       ('flag', 0)])),
                                         ('product_type_code', 1),
                                         ('scheduling_code', 2),
                                         ('seconds_between_runs', 0),
                                         ('generation_time',
                                          datetime.datetime(2020, 4, 1, 17, 36, 58, 858000)),
                                         ('sweep_ingest_time',
                                          datetime.datetime(2020, 4, 1, 17, 36, 5, 297000)),
                                         ('file_ingest_time',
                                          datetime.datetime(2020, 4, 1, 17, 36, 5, 297000)),
                                         ('product_name', 'Z_000_370   '),
                                         ('task_name', 'SURVEILLANCE'),
                                         ('flag', 0),
                                         ('x_scale', 102778),
                                         ('y_scale', 102778),
                                         ('z_scale', 0),
                                         ('x_size', 720),
                                         ('y_size', 720),
                                         ('z_size', 1),
                                         ('x_location', 359985),
                                         ('y_location', 360102),
                                         ('z_location', 0),
                                         ('maximum_range', 0),
                                         ('data_type', 2),
                                         ('projection_name', 'AE_WGS_370'),
                                         ('input_data_type', 2),
                                         ('projection_type', 0),
                                         ('radial_smoother', 0),
```

```
                                ('times_run', 12278),
                                ('zr_constant', 0),
                                ('zr_exponent', 0),
                                ('x_smoother', 100),
                                ('y_smoother', 100),
                                ('product_specific_info',
                                 OrderedDict([('elevation_angle',
                                               0.0)])),
                                ('minor_task_suffixes', ''),
                                ('color_scale_def',
                                 OrderedDict([('iflags', 0),
                                              ('istart', 0),
                                              ('istep', 0),
                                              ('icolcnt', 0),
                                              ('iset_and_scale', 0),
                                              ('ilevel_seams',
                                               array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
dtype=uint16))]))])),
                               ('product_end',
                                OrderedDict([('site_name', 'RPG7            '),
                                             ('iris_version_created', '8.13'),
                                             ('ingest_iris_version', '8.13'),
                                             ('ingest_time',
                                              datetime.datetime(2020, 4, 1, 17, 36, 5, 297000)),
                                             ('GMT_minute_offset_local', 0),
                                             ('ingest_hardware_name_',
                                              'Hatay Radar     '),
                                             ('ingest_site_name_',
                                              'Hatay Radar     '),
                                             ('GMT_minute_offset_standard', 0),
                                             ('latitude', 36.31806006655097),
                                             ('longitude', 35.78817006200552),
                                             ('ground_height', 304),
                                             ('radar_height', 23),
                                             ('prf', 400),
                                             ('pulse_width', 500),
                                             ('signal_processor_type', 8),
                                             ('trigger_rate', 0),
                                             ('samples_used', 32),
                                             ('clutter_filter', ''),
                                             ('number_linear_filter', 4),
                                             ('wavelength', 533),
                                             ('truncation_height', 2000000),
```

('first_bin_range', 0),
('last_bin_range', 37000000),
('number_bins', 493),
('flag', 0),
('number_ingest', 1),
('polarization', 3),
('horizontal_calibration_i0', -11495),
('horizontal_calibration_noise',
-8401),
('horizontal_radar_constant', 6009),
('receiver_bandwidth', 269),
('horizontal_current_noise', -8402),
('vertical_current_noise', -8489),
('ldr_offset', -147),
('zdr_offset', -112),
('tcf_cal_flags_1', 21),

('tcf_cal_flags_2', 5),

('standard_parallel_1', 0.0),

('standard_parallel_2', 0.0),

('earth_radius', 637813700),

('inverse_flatting', 298257224),

('fault_status', 1),

('input_mask', 0),

('number_log_filter', 0),

('cluttermap', 0),

('latitude_projection',

36.31806006655097),

('longitude_projection',

35.78817006200552),

('product_sequence_number', 0),

('melting_level', -30068),

('radar_height_above_reference', 327),

('number_elements', 0),

('mean_wind_speed', 0),

```
                    ('mean_wind_direction', 0.0),
                    ('tz_name', 'UTC'),
                    ('extended_product_header_offset',
                     0)]))])),
('product_type', 'PPI'),
('nbins', 493),
('data',
 OrderedDict([(0,
               array([[[95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5],
                       [95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5],
                       [95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5],
                       ...,
                       [95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5],
                       [95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5],
                       [95.5, 95.5, 95.5, ..., 95.5, 95.5, 95.5]]]))])])
```

Ek 8. Toplam Yağış Miktarı Verisi İstenen Meteoroloji İstasyonları

Istasyon Numarası	İstasyon Adı
17255	KAHRAMANMARAŞ
17256	KAHRAMANMARAŞ HAVALİMANI
17320	ANAMUR
17330	SİLİFKE
17340	MERSİN
17350	İNCİRLİK MEYDAN
17351	ADANA BOLGE
17352	ADANA SAKIRPASA HAVALIMANI
17355	OSMANIYE
17370	İSKENDERUN
17371	HATAY HAVALIMANI
17372	ANTAKYA
17645	HATAY TİGEM
17649	CEYHAN TİGEM
17866	GÖKSUN
17868	AFŞİN
17870	ELBİSTAN
17907	KADİRLİ
17908	KOZAN
17934	POZANTI
17936	KARAİSALI
17956	MUT
17958	ERDEMLİ
17960	CEYHAN
17962	DÖRTYOL
17965	ISLAHİYE
17978	TARSUS
17979	YUMURTALIK
17981	KARATAŞ
17986	SAMANDAĞ
18053	TUFANBEYLİ
18055	İMAMOĞLU
18056	SAİMBEYLİ
18057	BELEN
18058	ALTINÖZÜ
18059	GÜLNAR
18060	ÇAMLIYAYLA/MEYVECİLİK ARŞ. (TAGEM)
18063	BAHÇE

18064	DÜZİÇİ
18156	ANDIRIN
18157	PAZARCIK
18258	ALADAĞ
18268	ÇUKUROVA
18269	FEKE
18270	SARIÇAM
18274	ERZİN
18275	HASSA
18276	KIRIKHAN
18278	YAYLADAĞ
18279	ÇAĞLAYANCERİT
18280	EKİNÖZÜ
18281	NURHAK
18282	TÜRKOĞLU
18284	BOZYAZI
18287	TOROSLAR/ARSLANKÖY
18289	HASANBEYLİ
18290	TOPRAKKALE
8258	CAMLIYAYLA
18858	ARSUZ
18862	POZANTI/AKÇATEKİR
17486	İSKENDERUN KÖRFEZİ
19180	MEZİTLİ
19184	SUMBAS
19363	DEFNE

Ek 9. Meteoroloji Genel Müdürlüğü ile Yapılan Yazışmalar

27/04/2021-85931

T.C.
ÇUKUROVA ÜNİVERSİTESİ
Fen Bilimleri Enstitüsü Müdürlüğü

Sayı : E-66846985-302.05.01-85931 27/04/2021
Konu : Veri Talebi

UZAKTAN ALGILAMA VE COĞRAFİ BİLGİ SİSTEMLERİ
ANABİLİM DALI BAŞKANLIĞINA

Anabilim Dalınız Tezli Yüksek Lisans öğrencisi Alişan AKTAY'ın bilimsel çalışma izin başvurusu ile ilgili Çukurova Üniversitesi Öğrenci İşleri Daire Başkanlığı'ndan alınan 26.04.2021 tarih ve E.83952sayılı yazı ve eki ilişikte gönderilmiştir.

Bilgilerinizi ve gereğini rica ederim.

Prof.Dr. Mustafa GÖK
Enstitü Müdürü

Ek:Yazı (2 sayfa)

Evrak Tarih ve Sayısı: 26/04/2021-83952

T.C.
TARIM VE ORMAN BAKANLIĞI
Meteoroloji Genel Müdürlüğü
Meteorolojik Veri İşlem Dairesi Başkanlığı

Sayı : E-95579059-107-34177 21.04.2021

Konu : Rasat Bilgisi ve Bilgi İstekleri

ÇUKUROVA ÜNİVERSİTESİ REKTÖRLÜĞÜNE
(Öğrenci İşleri Daire Başkanlığı)
Balcalı Mah. Remzioğuzarık
01270 Sarıçam / ADANA

İlgi : 14.04.2021 tarihli ve E-27224817-044-77797 sayılı ve 12810 Belgenet kayıt nolu yazınız.

İlgi yazı ile istenilen bilgiler, ilgili meteoroloji istasyonlarımızın mevcut kayıtlarından çıkartılarak ekte verilen bağlantı adresinde sunulmuştur. Radar verileri arşivine ulaşım izni yalnızca Genel Müdürlük ilgili birimlerine tanımlanmakta olup ayrıca istenmesi halinde radar görüntülerinin yeni bir yazı ile ilgili birimlerden talep edilmesi mümkündür.

10 Haziran 2020 tarihli ve 31151 sayılı Resmi Gazete'de yayımlanan "Resmi Yazışmalarda Uygulanacak Usul ve Esaslar Hakkında Yönetmelik" uyarınca resmi yazışmalarda paylaşım saklama işlemleri için elektronik ortam kullanımı esas unsur haline getirilmiştir. Bu kapsamda istemiş olduğunuz meteorolojik veriler, yazımızın tarihi itibariyle 3 ay süresince verilen ekteki bağlantı üzerinden indirilebilecektir.

Bilgilerinize arz ederim.

Selami YILDIRIM
Genel Müdür a.
Meteorolojik Veri İşlem Dairesi Başkanı

Ek: https://dosya.tarimorman.gov.tr/app/tr-TR/[redacted]

Evrak Tarih ve Sayısı: 07/07/2021-135310

T.C.
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜ
Uzaktan Algılama ve Coğrafi Bilgi Sistemleri Anabilim Dalı Başkanlığı

Sayı : E-31713732-399-135310 07/07/2021
Konu : Dr. Öğr. Üyesi Nuri EMRAHOĞLU'nun Veri Talebi (Meteoroloji Genel Müdürlüğü)

FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE

Anabilim Dalımız Öğretim Üyesi Dr. Öğr. Üyesi Nuri EMRAHOĞLU'nun danışmanlığını yaptığı Yüksek Lisans Öğrencisi Alişan AKTAY'ın "Meteoroloji Radar Görüntüleri Kullanılarak Oluşabilecek Bir Sonraki Görüntünün Yapay Zeka ile Tahmini: Hatay Radar Örneği" başlıklı yüksek lisans tez çalışması kapsamında ihtiyaç duyulan veriler ile ilgili 06.07.2021 tarihli talep dilekçesi ekte sunulmuştur.
Gereğini bilgilerinize arz ederim.

Prof.Dr. Süha BERBEROĞLU
Anabilim Dalı Başkanı

Ek:Dilekçe