RC11 SKILLS FINAL REPORT

Research Cluster 11
Student Number 23188508

INFO

Supporting folders are available on OneDrive via the following link:
https://liveuclac-my.sharepoint.com/:f:/g/personal/ucbvmmk_ucl_ac_uk/Ev4Bouh6C5FFmJ5eFS9eXxMBWz7CiNbFmqUhqtVEyNdaBw?e=XfU3IV

CONTENT

VECTORIAL ENCODINGS OF QUALITATIVE DOMAINS

I. DATA COLLECTION

INTRODUCTION

Classism

What is a classic design? The inquiry about the “Classic” begins with defining it using adequate terminology. As part of the development of the Search Engine, this introduction presents a nuanced outlining for directing later steps and decisions in the project. Looking at the “classic” from a critical point of view, Demetri Porphyrios presents in his paper (Classisim is not a style) a critical rethinking of what is the classic. He argues that classicism should not be reduced to merely a stylistic preference or a set of visual characteristics (Amundson, 2015). Instead, he states that classicism is a comprehensive philosophical and cultural framework that encompasses principles of proportion, harmony, order, and continuity. This framework is transformed into a physical entity by the constructional logic of vernacular. In conclusion, he emphasizes the importance of recognizing classicism as a holistic approach to architecture and design, rather than merely as a superficial style.

Columns Encyclopedia

In the context of columns, based on the earlier introduction it would be reductive to state that there are three main orders of columns: Doric, Ionic, and Corinthian. “Heterodoxia Architectonica” is a research by Andrés Duany that emphasizes the departure from the classical conventional understanding of columns (Congress for the New Urbanism, 2012). Each column is worthy of documentation and presents a style by itself. He criticizes the perception of classicism, although he aligns with Porphyrios but both of them did not build their arguments based on each other’s work.

Data Collection

The collected data presents the reflection of the discussed logic. The datasets form the ground to conduct an analytical exercise whereby an array of sources are consulted, critically reinterpreted, and reformulated to begin to arrive at the essence of each column. Given the nature of the columns, the datasets consist of text, images, videos, and text 02. These datasets resonate with the earlier introduction. Each dataset is explained in detail in the following pages.

DATASET 01: TEXT

The first text dataset consists of 983 webscraped architectural books that contain 39,779 sentences associated with columns. The architectural books include books such as A Pattern Language by Christopher Alexander, Elements of Architecture by Rem Koolhaas, and Ten Books on Architecture by Vitruvius. These books will provide a dictionary of definitions and descriptions of any hypothetical column. They represent the existing architectural knowledge on columns that can be used to sculpt new interpretations.

Folder_Path = “Datasets/ Text/ Books”

Abulafia__The_Boundless_Sea.txt
Ackerman__Origins_Imitation_Conventions_Representation_in_the_Visual_Arts.txt
Ackerman__The_Architecture_of_Michelangelo.txt
Acocella__Stone_Architecture_Ancient_and_Modern_Construction_Skills.txt
Adams__Empire_and_Communications.txt
Adams__Mont_Saint_Michel_and_Chartres.txt
Adam__Ruins_of_the_Palace_of_the_Emperor_Diocletian_at_Spalatro_in_Dalmatia.txt
Adorno__The_Stars_Down_to_Earth_and_Other_Essays_on_the_Irrational_in_Culture.txt
Aeschylus__The_Persians.txt
Agrest_Conwy_Weisman__The_Sex_of_Architecture.txt
Agricola_De_Re_Metallica.txt
Agrippa__The_Fourth_Book_of_Occult_Philosophy.txt
Agrippa__Three_Books_of_Occult_Philosophy.txt
Aharoni__Mathematics_Poetry_and_Beauty.txt
Alberti__10_books_of_architecture.txt
Alberti__Delineation_of_the_City_of_Rome.txt
Alberti__Momus.txt
Alberti__On_Painting.txt
Alberti__On_the_Art_of_Building_in_Ten_Books.txt
Alberti__The_Mathematical_Works_of_Leon_Battista_Alberti.txt
Alder__Engineering_the_Revolution.txt
Alexander__A_Pattern_Language.txt
Alexander__From_Renaissance_to_Counter_Reformation.txt
Alexander__The_Timeless_Way_of_Building.txt
Allatios__The_Newer_Temples_of_the_Greeks.txt
Anderson__Antiquities_What_Everyone_Needs_to_Know.txt
An__Atlas_of_Fantastic_Infrastructures.txt
Aquinas__Summa_Theologica.txt
Archimedes__On_the_Sphere_and_the_Cylinder.txt
Arendt__Eichmann_in_Jerusalem.txt
Aristotle__A_Treatise_on_Government.txt
Aristotle__Ethics.txt
Aristotle__Metaphysics.txt
Aristotle__Physics.txt
Aristotle__Politics.txt
Aristotle__The_Art_of_Rhetoric.txt
Arnold__Literature_and_Dogma.txt
ArtBasel__Catalogue.txt
Ascher__The_Works_Anatomy_of_a_City.txt
Asimov__Complete_Robot_Anthology.txt
Asimov__I_Robot.txt
Askin__Narrative_and_Becoming.txt
Aureli__The_Possibility_of_an_Absolute_Architecture.txt
Avanessian_Hennig__Present_Tense_A_Poetics.txt
Avery_Finn__Bernini.txt
Ayache__The_Blank_Swan.txt
Bacchi__Bernini_and_the_Birth_of_Baroque_Portrait_sculpture.txt
Bacon__Selected_Philosophical_Works.txt
Bacon__The_Advancement_of_Learning.txt
Bacon__The_New_Atlantis.txt
Baldinucci__The_Life_of_Bernini.txt
Ball__The_Domus_Aurea_and_the_Roman_Architectural_Revolution.txt
Ball__The_Selfmade_Tapestry_Pattern_Formation_in_Nature.txt
Balzac__The_Unknown_Masterpiece.txt
Banham__Critic_Writes.txt
Banham__Theory_and_Design_in_the_First_Machine_Age.txt
Banham__The_Architecture_of_the_Well_Tempered_Enviroment.txt
Bann__The_Clothing_of_Clio.txt
Barasch__Theories_of_Art.txt
Barber__A_Companion_To_World_Mythology.txt
Barnes__Michelangelo_in_Print.txt
Barry__The_Rome_of_Piranesi.txt
Barthes__Empire_of_Signs.txt
Barthes__Mythologies.txt
Barthes__The_Fashion_System.txt
Barthes__The_Language_of_Fashion.txt
Barthes__Writing_Degree_Zero.txt
Bataille__%20Death_and_Sensuality.txt
Bataille__Death_and_Sensuality_A_Study_of_Eroticism_and_the_Taboo.txt
Baudrillard_Nouvel__The_Singular_Objects_of_Architecture.txt
Baudrillard__Simulacra_and_Simulation.txt
Baxandall__Giotto_and_the_Orators.txt
Bayle_Bartlett__Various_Thoughts_on_the_Occasion_of_a_Comet.txt
Beauty__Poems_From_Plenty.txt
Beckett__Stories_and_Texts_for_Nothing.txt
Belier_Bergdoll_Lecoeur__Henri_Labrouste_Structure_Brought_to_Light.txt
Bell__Art_History_in_the_Age_of_Bellori.txt
Bell__Men_of_Mathematics.txt
Belting__Florence_and_Baghdad.txt
Belting__Likeness_and_Presence.txt
Benjamin__The_Work_of_Art_in_the_Age_of_M.txt
Bennett__Influx_and_Efflux.txt
Bennett__The_Mathematical_Science_of_Christopher_Wren.txt
Benton_Sharp__Form_and_Function.txt
Bergdoll_Oechslin__Fragments_Architecture_and_the_Unfinished.txt
Berkenhout__The_Ruins_of_Poestum_or_Posidonia_a_City_of_Magna_Graecia_in_the_Kingdom_of_Naples.txt
Bernini__The_Life_of_Gian_Lorenzo_Bernini.txt
Bernoulli__The_Art_of_Conjecturing.txt
Betsky__Architecture_Matters.txt
Bichat__General_Anatomy_Applied_to_Physiology_and_Medicine_Vol_1.txt
Bichat__General_Anatomy_Applied_to_Physiology_and_Medicine_Vol_2.txt
Bichat__General_Anatomy_Applied_to_Physiology_and_Medicine_Vol_3.txt
Biebricher__The_Political_Theory_of_Neoliberalism.txt
Blacklock__The_Emergence_of_the_Fourth_Dimension.txt
Blackwell__Nineteenth_Century_Architecture.txt
Black__Experiments_upon_Magnesia_Alba_Quicklime_and_some_other_Alcaline_Substances.txt
Blonde__City_and_Society_in_the_Low_Countries_1100_1600.txt
Blunt__Artistic_Theory_in_Italy_1450_1600.txt
Blunt__Art_and_Architecture_in_France_1500_1700.txt
Blunt__Baroque_and_Rococo_Architecture_and_Decoration.txt
Blunt__Guide_to_Baroque_Rome.txt
Blunt__Philibert_de_lOrme.txt
BoBardi__Stones_Against_Diamonds.txt
Bohnet__What_Works.txt
Bolzoni__The_Gallery_of_Memory.txt
Bonnemaison_Macy__Festival_Architecture.txt
Bonnet__A_Compendium_of_Natural_Philosophy.txt
Boole__The_Laws_of_Thought.txt
Borges__Collected_Fictions.txt
Bork__Late_Gothic_Architecture.txt
Borromeo__Sacred_Painting.txt
Bosker__Original_Copies.txt
Bottazzi__Digital_Architecture_Beyond_Computers_Fragments_of_a_Cultural_History_of_Computational_Design.txt
Boucher__Italian_Baroque_Sculpture.txt
Boullee__Architecture_Essay_on_Art.txt
Bourdieu__Distinction.txt
Boyd__The_Australian_Ugliness.txt
Bradbury__Romantic_Theories_of_Architecure_of_the_19th_century.txt
Bradley__Smell_and_the_Ancient_Senses.txt
Braidotti_Dolphijn__Philosophy_After_Nature.txt
Braidotti__Patterns_of_Dissonance.txt
Bray__The_Sacred_Made_Real.txt
Bredekamp__The_Lure_of_Antiquity_and_the_Cult_of_the_Machine.txt
Brillouin__Science_and_Information_Theory.txt
Brook__A_History_of_Future_Cities.txt
Brown__The_Genius_of_Rome_1592_1623.txt
Bruno__On_the_Composition_of_Images_Signs_and_Ideas.txt
Buehlmann_Hovestadt__Coding_as_Literacy.txt
Buehlmann_Hovestadt__Domesticating_Symbols.txt
Buehlmann_Hovestadt__Symbolizing_Existence.txt
Buehlmann__Mathematics_and_Information_in_the_Philosophy_of_Michel_Serres.txt
Buffon__Natural_History_Quadrupeds.txt
Buffon__Natural_History_Vol_1.txt
Buffon__Natural_History_Vol_2.txt
Buffon__Natural_History_Vol_4.txt
Buffon__Natural_History_Vol_6.txt
Buffon__Natural_History_Vol_7.txt
Buffon__Natural_History_Vol_8.txt
Burckhardt__The_Architecture_of_the_Italian_Renaissance.txt
Burke__Art_And_Identity_In_Early_Modern_Rome.txt
Burke__Hybrid_Renaissance.txt
Burke__Rethinking_the_High_Renaissance.txt
Burke__The_Fabrication_of_Louis_XIV.txt
Burrows__Fictioning.txt
Bussels__Spectacle_Rhetoric_and_Power.txt
Butler__Synaesthesia_and_the_Ancient_Senses.txt
Cache__Projectiles.txt
Cahan__They_All_Fall_Down.txt
Calasso__Ardor.txt
Calasso__Ka_Stories_of_the_Mind_and_Gods_of_India.txt
Calasso__The_Marriage_of_Cadmus_and_Harmony.txt
Callan__Dictionary_of_Fashion_and_Fashion_Designers.txt
Calvino__Invisible_Cities.txt
Calvin__Harmony_of_the_Law_Vol_1.txt
Calvin__Harmony_of_the_Law_Vol_2.txt
Calvin__Harmony_of_the_Law_Vol_3.txt
Calvin__Harmony_of_the_Law_Vol_4.txt
Cameron__The_Baths_of_the_Romans_Explained_and_Illustrated.txt
Campanella__The_Book_and_the_Body_of_Nature.txt
Campanella__The_City_of_the_Sun.txt
Camus__The_Myth_Of_Sisyphus_And_Other_Essays.txt
Carpo_Furlan__Leon_Battista_Albertis_Delineation_of_the_City_of_Rome.txt
Carpo__Architecture_in_the_Age_of_Printing.txt
Carpo__The_Alphabet_and_the_Algorithm.txt

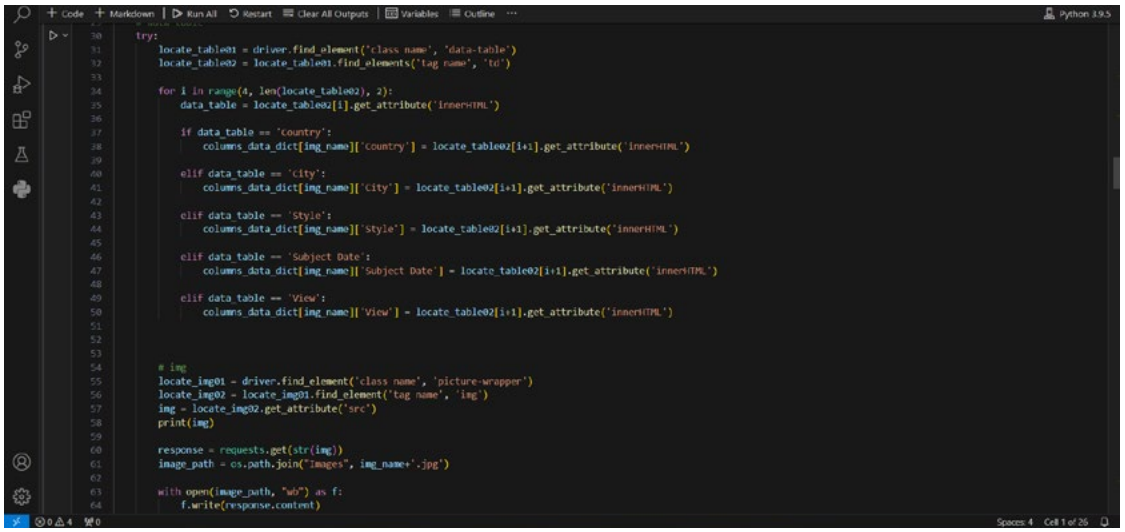
DATASET 02: IMAGES

The second dataset consists of 4411 webscraped images of columns. The locations of the images are diverse including the UK, Italy, Greece, USA, France, etc. The information of each image is stored in a dictionary with the keys of 'name', 'Country', 'City', 'Style', 'Subject Date', and 'View'.

Sample from the dictionary:
{"Great-Bath-Bath": {"Country": "UK: England", "City": "Bath", "Subject Date": "0001", "View": "Exterior", "Style": "Roman"}, "Rotunda-Wentworth-Castle-South-Yorkshire-detail-of-entablature": {"Country": "UK: England", "Subject Date": "1742", "View": "Exterior", "Style": "Palladian"}, "Scarborough-North-Yorkshire-aerial-perspective-of-The-Crescent-and-surrounding-buildings-arranged-around-a-circular-green-and-column": {"Country": "UK: England", "City": "Scarborough", "Subject Date": "1832", "View": "Aerial", "Style": "Classical Revival"}

Then all of this information all combined in a new key named 'info' to be used later.

Sample of the list:
[
{
"Subject Date": "0001",
"name": "0_Great-Bath-Bath.jpg",
"info": "UK: England, Bath, Roman, Exterior, Great Bath Bath"
},
{
"Subject Date": "1742",
"name": "1_Rotunda-Wentworth-Castle-South-Yorkshire-detail-of-entablature.jpg",
"info": "UK: England, Palladian, Exterior, Rotunda Wentworth Castle South Yorkshire detail of entablature"
},



```
try:
    locate_table01 = driver.find_element('class name', 'data-table')
    locate_table02 = locate_table01.find_elements('tag name', 'td')

    for i in range(4, len(locate_table02), 2):
        data_table = locate_table02[i].get_attribute('innerHTML')

        if data_table == 'Country':
            columns_data_dict[img_name]['Country'] = locate_table02[i+1].get_attribute('innerHTML')

        elif data_table == 'City':
            columns_data_dict[img_name]['City'] = locate_table02[i+1].get_attribute('innerHTML')

        elif data_table == 'Style':
            columns_data_dict[img_name]['Style'] = locate_table02[i+1].get_attribute('innerHTML')

        elif data_table == 'Subject Date':
            columns_data_dict[img_name]['Subject Date'] = locate_table02[i+1].get_attribute('innerHTML')

        elif data_table == 'View':
            columns_data_dict[img_name]['View'] = locate_table02[i+1].get_attribute('innerHTML')



    # img
    locate_img01 = driver.find_element('class name', 'picture-wrapper')
    locate_img02 = locate_img01.find_element('tag name', 'img')
    img = locate_img02.get_attribute('src')
    print(img)

    response = requests.get(str(img))
    image_path = os.path.join("Images", img_name+'.jpg')

    with open(image_path, "wb") as f:
        f.write(response.content)
```

* Webscraping code snippet

Data Sample

Folder_Path = “Datasets/ Images/ Images”

{“name”: “0_Great-Bath-Bath.jpg”, “info”: “Exterior, Roman, Bath, UK: England, 0001, Great Bath Bath”}

{“name”: “1_Rotunda-Wentworth-Castle-South-Yorkshire-detail-of-entablature.jpg”, “info”: “Exterior, Palladian, UK: England, 1742, Rotunda Wentworth Castle South Yorkshire detail of entablature”}

{“name”: “2_Scarborough-North-Yorkshire-aerial-perspective-of-The-Crescent-and-surrounding-buildings-arranged-around-a-circular-green-and-column.jpg”, “info”: “Aerial, Classical Revival, Scarborough, UK: England, 1832, Scarborough North Yorkshire aerial perspective of The Crescent and surrounding buildings arranged around a circular green and column”}

{“name”: “3_Royal-Brunswick-Theatre-Wellclose-Square-Stepney-London-street-elevation.jpg”, “info”: “Exterior, Greek Revival, London, UK: England, 1828, Royal Brunswick Theatre Wellclose Square Stepney London street elevation”}

{“name”: “4_St-Philips-Chapel-Regent-Street-London-plan-and-west-elevation.jpg”, “info”: “Greek Revival, London, UK: England, 1820, St Philips Chapel Regent Street London plan and west elevation”}

{“name”: “5_Heathcote-Ilkley-West-Yorkshire-the-main-hall.jpg”, “info”: “Interior, Edwardian, 1906, Heathcote Ilkley West Yorkshire the main hall”}

{“name”: “6_Heathcote-Ilkley-West-Yorkshire-the-lobby-off-the-main-hall.jpg”, “info”: “Interior, Edwardian, Ilkley, UK: England, 1906, Heathcote Ilkley West Yorkshire the lobby off the main hall”}

{“name”: “7_Plaza-de-Espana-from-the-Ibero-American-Exhibition-of-1929-Maria-Luisa-Park-Seville-the-tiled-alcoves.jpg”, “info”: “Exterior, Seville, Spain, 1929, Plaza de Espana from the Ibero American Exhibition of 1929 Maria Luisa Park Seville the tiled alcoves”}

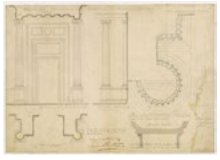
{“name”: “8_National-Building-Museum-4th-and-F-Streets-Washington-DC-the-main-hall.jpg”, “info”: “Interior, Washington, D.C., United States, 1887, National Building Museum 4th and F Streets Washington DC the main hall”}

{“name”: “9_St-Leonards-on-Sea-East-Sussex-view-of-the-west-end-of-the-marina-Victoria-House-and-St-Leonards-Church.jpg”, “info”: “Exterior, St Leonards on Sea, UK: England, 1830, St Leonards on Sea East Sussex view of the west end of the marina Victoria House and St Leonards Church”}

{“name”: “10_St-Leonards-on-Sea-East-Sussex-view-of-the-west-end-of-the-marina-and-Victoria-House.jpg”, “info”: “Exterior, St Leonards on Sea, UK: England, 1830, St Leonards on Sea East Sussex view of the west end of the marina and Victoria House”}

{“name”: “11_Coilsfield-or-Montgomerie-House-near-Tarbolton-Ayrshire-perspective-from-the-park.jpg”, “info”: “Exterior, Classical Revival, Tarbolton, UK: Scotland, 1798, Coilsfield or Montgomerie House near Tarbolton Ayrshire perspective from the park”}

{“name”: “12_Detail-of-capital-possibly-from-Palazzo-Strozzi-Bevilacqua-or-Palazzo-Rondinelli-piazza-Ariostea-Ferrara.jpg”, “info”: “Worm’s-eye, Renaissance, Ferrara, Italy, 1400, Detail of capital possibly from Palazzo Strozzi Bevilacqua or Palazzo Rondinelli piazza Ariostea Ferrara”}



{“name”: “13_Designs-for-the-Reform-Club-Pall-Mall-London-elevation-and-details-of-columns-and-pilasters-in-alterations-to-coffee-room.jpg”, “info”: “Interior, Italianate, London, UK: England, 1853, Designs for the Reform Club Pall Mall London elevation and details of columns and pilasters in alterations to coffee room”}

{“name”: “14_The-Circus-Bath-detail-of-the-facade.jpg”, “info”: “Exterior, Georgian, Bath, UK: England, 1766, The Circus Bath detail of the facade”}

{“name”: “15_Eastern-Dispensary-Cleveland-Place-East-Bath.jpg”, “info”: “Exterior, Classical Revival, Bath, UK: England, 1845, Eastern Dispensary Cleveland Place East Bath”}

DATASET 03: VIDEOS

The third dataset consists of 135 web scraped videos showcasing different places in London. London was chosen as it is the most frequent location in the image dataset. The videos document various locations within Greater London. Information for each video is stored in a dictionary with keys of ‘name’, ‘summary’, ‘description’, and ‘style’.

* datasets used for training the columns detection model are availbe on:
File_Path = Datasets/Videos/ Columns Detection model/ Images

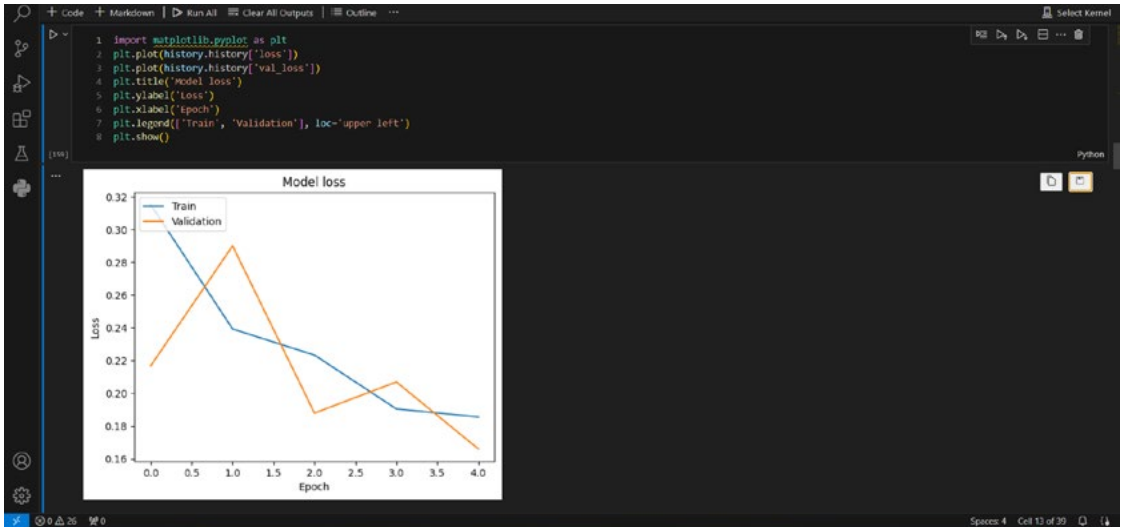
Columns Detection Model

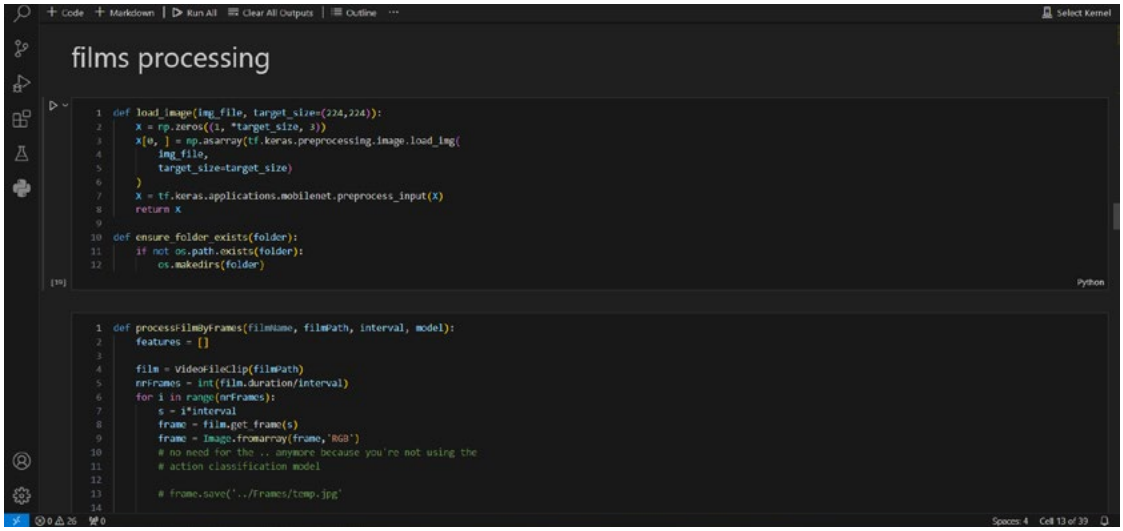
As part of the video processing pipeline, an image classification model was trained specifically to identify column scenes within the video dataset. The model was trained using five distinct types of images: buttresses, columns, domes, vaults, and random images. Subsequently, I used the column detection model to analyze all the videos and generate a list of the ten-second that most accurately correspond to column scenes. This list is then merged with the existing dictionary.

The process of training the model starts by inputting the 5 different categories of images:

1. Train the model and save it.
2. Extract the features of the videos.
3. Sort the features based on proximity to the category “column”.
4. Extract the seconds of the closet clips.

* Column detectoin model code snippets





Data Sample

Folder_Path = “Datasets/ Videos/ columns_videos_data.json”

Final list strucutre

{“name”: “Lights of London”,
“summary”: “ Night footage including Christmas lights, stores and tourist sights taken of Central London by amateur filmmaker Bernard Chaplin.”,
“description”: “ The film shows the Christmas lights around Oxford Street and Regent Street and the colourful advertising around Piccadilly Circus. Red and green garlands of lights hang across the street under which vehicles pass. There are glimpses of several stores – Hamleys, Galeries Lafayette, Burberrys and Swan and Edgar. Illuminated advertising hoardings light up Piccadilly Circus. Adverts for alcohol, cigarettes and chewing gum dominate the signs. An illuminated sign for the Regent Place Hotel can be seen. The film finishes with a Christmas tree decorated with coloured lights.”,
“second”: [54, 0, 132, 36, 48, 108, 126, 114, 42, 12]},

{“name”: “Local views of Friern Barnet”,
“summary”: “ Local footage from the Friern Barnet area”,
“description”: “ The film starts with the street sign Lawrence Campe Close, N20, on a white picket fence. It then shows a new building, possibly new almshouses around a lawned courtyard, behind a terrace of old buildings. The gravel driveway to the new building has then got a crowd of people, clergy and a bishop all in formal dress opening the building. The clergy give a speech regarding the building and the film cuts to a shot of a plaque, this reads \”These Almshouses blessed by the Bishop of London on 2nd Oct 1984\”.\n”,
“second”: [42, 36, 156, 570, 636, 210, 198, 162, 30, 270] },

{“name”: “Cyril Flat”,
“summary”: “ Different views from and of a flat in Sara House, Larner Road, Erith.”,
“description”: “ Outside Flats outside flats overlooking pub on corner. Sara House taken from road. More shots of Sara House. Now up in flat looking down. More views of urban planning. Older lady sitting under parasol in garden - underexposed. Garden views. Possible shots of grandchildren.”,
“second”: [72, 42, 12, 36, 192, 180, 126, 0, 96, 84] },

Data Sample

Folder_Path = “Datasets/ Videos/Clips”

A Good Name The Story of One Neighbourhood.mp4
A Journey Through Time.mp4
About Hampstead.mp4
Albany Mission and construction of Islington Central Methodist Church.mp4
Alfred Sansom and Family.mp4
Alice and Andrew Orpington.mp4
All On A Winters Day.mp4
Around amp About Bexley.mp4
Around Kingston 50s amp 60s.mp4
Around Richmond 73 79 1980.mp4
Around Stanwell.mp4
Barbican Regained.mp4
Beauty in the Borough.mp4
Bexleyheath Broadway Past and Present.mp4
Bookfair 89 Peckham Bookplace Bookfair Part 1.mp4
Brent A State of Mind.mp4
Bruce Shawcroft Films FebruaryMarchApril 1970 June 1971.mp4
Bruce Shawcroft Films MarchApril 1970.mp4
Camberwell Road.mp4
Christmas in Newham.mp4
Clubland Activities of the 1950s and 60s.mp4
Cranford Park and St Dunstans Church.mp4
Croydon in the 1970s Part 1.mp4
Croydon in the 1980s Part 2.mp4
Cyril Flat.mp4
Cyril Visits Lesney Farm Allotment and Pubs in Erith.mp4
Danson Park.mp4
Development of Romford in the Sixties.mp4
Discovering Newham.mp4
Dogs Life 3.mp4
Dogs Life Issue One.mp4
Dublin and District.mp4
Duckers n Divas.mp4
East Wickham and Welling.mp4
Erith Town Centre.mp4
Erith Town.mp4
Family together including Tottenham carnival 1964.mp4
Filming the Unusual.mp4
For What We Are About To Lose Carlton Mansions.mp4
Frequency 16.mp4
Georges Square Hoxton.mp4
Growing Up in Southwark Community Life in Southwark and Passport to the Future.mp4
Hackney Housing Estates.mp4
Hampstead Old amp New.mp4
Harrow Heritage.mp4
Hayes and Harlington.mp4
Here and There 1.mp4
High Hopes Voices from the Streets of Lewisham.mp4
Hounslow High St and Staines Road.mp4
If Walls Could Speak.mp4
In the Counties Around amp About Kent.mp4
Its Our Park a film about Southwark Park in the Summer of 2014.mp4
Kew Gardens during the War.mp4
Kingston in Decay.mp4
Kingston Movie Makers History of Walton.mp4
Lambeth c1972.mp4
Life of the Bicycle.mp4
Lights of London.mp4
Local Lore Just around the Corner Around amp About Bexley.mp4
Local Lore Now and Then.mp4
Local Lore Out for a Walk Around amp About Bexley.mp4
Local views of Friern Barnet.mp4
London Albert amp George Docks Chusan Wharf.mp4
London Albert Dock Dry Dock.mp4
London Borough.mp4
London Bridge.mp4
London Looking at London London No 6 Into the 70s.mp4
London Petticoat Lane Barrels Tower Bridge Docks Pros amp Whit Wapping.mp4
London Scenes c 1960.mp4
London Scenes Tower.mp4
London Sid amp Dave Hello amp Goodbye Guards at Bank Piccadilly Trafalgar Square Mall.mp4
London Sid Buck House Embankment.mp4
London Something Old Something New.mp4
London St Katherines Dock Tower Bridge Cable St.mp4
Looking for Sierra Leone.mp4
Loyalty amp Service Jimmy Butterworth amp Clubland.mp4
Manor Park.mp4
My Elephant.mp4
Newham in Widescreen.mp4
Notting Hill reel 1.mp4
Notting Hill reel 2.mp4
Notting Hill reel 3.mp4
Our First Film .mp4
Oxford Street.mp4
Piccadilly Circus Timelapse.mp4
Potter Allotment Lesney Farm.mp4
Pullenites.mp4
Richmond 1 19551968.mp4
Rickmansworth Week 1956.mp4
Royal Borough.mp4
Sample
Save The Rose.mp4
SE18 Impressions of a London Suburb.mp4
South Gate California.mp4
Southgate Tech From the Air.mp4
Southwark Cities series People and Planners.mp4
Southwarks Pride.mp4
Spring in Hackney.mp4
St Olaves on Horsleydown.mp4
Stately Homes of Newham.mp4
Street Scene Muswell Hill II.mp4
Street Scene.mp4
Sum of Us All Bede House Bermondsey and Rotherhithe 19382008.mp4
Swan Corner.mp4
Taken from East Lane Bridge.mp4
Thamesmead.mp4
The Castle.mp4
The Changing Face of Camberwell.mp4
The Development of Marks Gate Housing Estate.mp4
The Early History of the Isle of Dogs.mp4
The Elephant Never Forgets.mp4
The Great West Road and Bath Road Near Hounslow.mp4
The New Croydon.mp4
The New North Heath Library Snows of 1991.mp4
The Spike a place of peace for restless minds.mp4
The Tarn Eltham.mp4
The Thames Down River.mp4
The Thames Londons River.mp4
The Thames Somethings Going On.mp4
The Tower.mp4
The Urban District of Hayes and Harlington.mp4
The Villages of Stepney.mp4
This is London.mp4
Thompson Home Movie Grove Hotel Ickenham and the Hillingdon Show.mp4
Tour of RuislipNorthwood UDC.mp4
Upminsters Heritage.mp4
Uxbridge Redevelopment.mp4
Uxbridge.mp4
Various Scenes including Metropolitan Borough of Hackney Housing Estates.mp4
Views Outside Hackney Town Hall with Warwick Grove Flats.mp4
Ville de Sceaux.mp4
Welcome to Thamesmead.mp4
WELLBELOVED.mp4
What Future for Haringey.mp4
Where and What Is Paradise.mp4
Worthy Inheritance.mp4

DATASET 04: TEXT 02

This study is not merely a technical study on columns, thus, to add a layer to the understanding from a different dimension, 94 fiction books were collected. This dataset will expand the understanding of columns beyond architectural interpretations. It will initiate a poetic conversation that is linked to columns in a nuanced approach. It adds a layer to understanding a column that is not merely technical

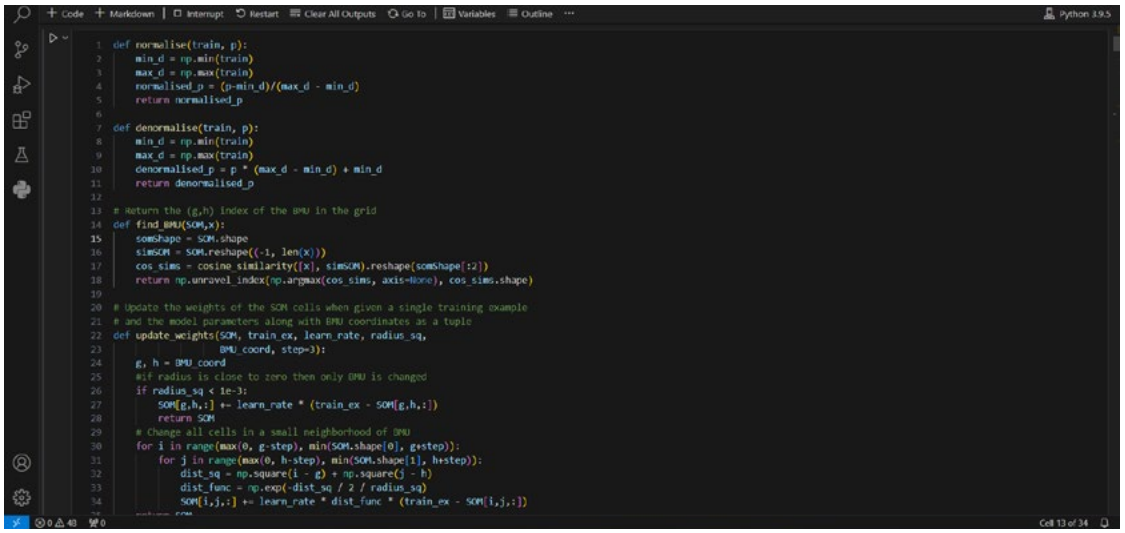
Folder_Path = “Datasets/ Text 02/ Books02”

pg10002.epub
pg10041.epub
pg10148-images.epub
pg10662.epub
pg10745-images.epub
pg1076-images.epub
pg1077-images.epub
pg10806.epub
pg10882-images.epub
pg11097.epub
pg11283.epub
pg11440.epub
pg1152.epub
pg11639-images.epub
pg11752.epub
pg1251.epub
pg1252.epub
pg1267-images.epub
pg12747.epub
pg12753-images.epub
pg1311.epub
pg13486.epub
pg13664.epub
pg13820.epub
pg13821.epub
pg15551-images.epub
pg1557.epub
pg15948.epub
pg1605-images.epub
pg16259-images.epub
pg16435.epub
pg169.epub
pg17134.epub
pg1751.epub
pg17973.epub
pg18328.epub
pg19393.epub
pg19959-images.epub
pg19973-images.epub
pg19976-images.epub
pg2060-images.epub
pg22566-images.epub
pg234-images.epub
pg2414.epub
pg2565.epub
pg2865.epub
pg288.epub
pg2885.epub
pg2892-images.epub
pg3055.epub
pg3261.epub
pg3687.epub
pg419.epub
pg420.epub
pg4282.epub
pg4356.epub
pg4358.epub
pg436.epub
pg485-images.epub
pg486.epub
pg5160.epub
pg517.epub
pg518.epub
pg54-images.epub
pg55-images.epub
pg5660.epub
pg5713-images.epub
pg5988-images.epub
pg604.epub
pg6050.epub
pg6582-images.epub
pg7477.epub
pg7838.epub
pg8129.epub
pg8183-images.epub
pg831.epub
pg832.epub
pg8395-images.epub
pg8715.epub
pg8771.epub
pg8778.epub
pg9488.epub
pg955.epub
pg956.epub
pg957.epub
pg958.epub
pg959-images.epub
pg960.epub
pg9608.epub
pg961.epub
pg964.epub
pg9663.epub
pg9829.epub

II. SELF-ORGANIZING MAPS

TEXT

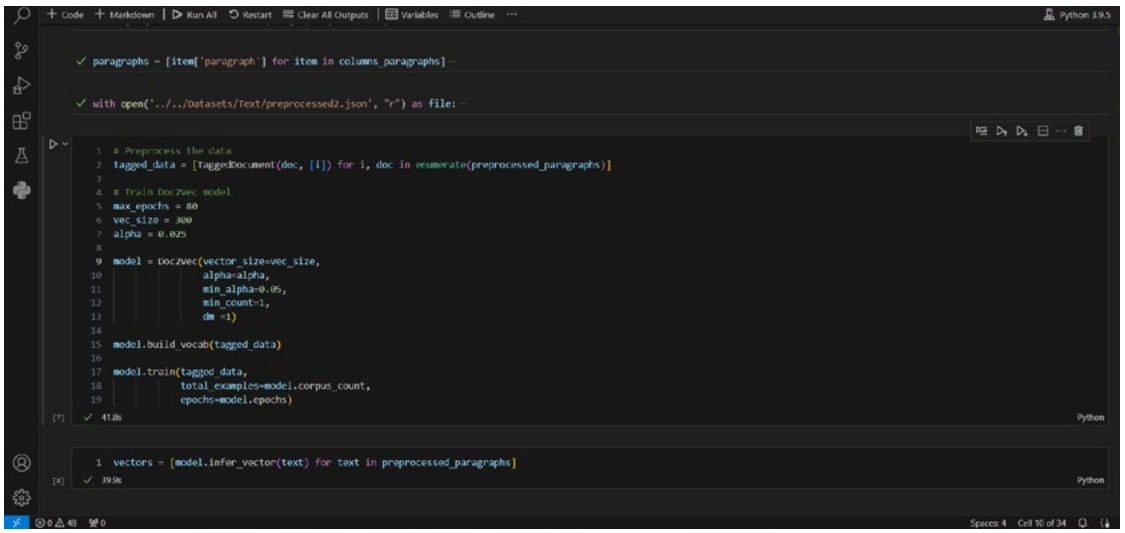
1. The process of developing a self-organizing map for text data starts by defining the functions. A key note is that when processing text cosine_similarity should be instead of Euclidean distance.



```python
def normalise(train, p):
    min_d = np.min(train)
    max_d = np.max(train)
    normalised_p = (p-min_d)/(max_d - min_d)
    return normalised_p

def denormalise(train, p):
    min_d = np.min(train)
    max_d = np.max(train)
    denormalised_p = p * (max_d - min_d) + min_d
    return denormalised_p

# Return the (g,h) index of the BMU in the grid
def find_BMU(SOM,x):
    somShape = SOM.shape
    simSOM = SOM.reshape((-1, len(x)))
    cos_sims = cosine_similarity([x], simSOM).reshape(somShape[:2])
    return np.unravel_index(np.argmax(cos_sims, axis=None), cos_sims.shape)

# Update the weights of the SOM cells when given a single training example
# and the model parameters along with BMU coordinates as a tuple
def update_weights(SOM, train_ex, learn_rate, radius_sq,
                   BMU_coord, step=3):
    g, h = BMU_coord
    #if radius is close to zero then only BMU is changed
    if radius_sq < 1e-3:
        SOM[g,h,:] += learn_rate * (train_ex - SOM[g,h,:])
        return SOM
    # Change all cells in a small neighborhood of BMU
    for i in range(max(0, g-step), min(SOM.shape[0], g+step)):
        for j in range(max(0, h-step), min(SOM.shape[1], h+step)):
            dist_sq = np.square(i - g) + np.square(j - h)
            dist_func = np.exp(-dist_sq / 2 / radius_sq)
            SOM[i,j,:] += learn_rate * dist_func * (train_ex - SOM[i,j,:])
```

* Defining necessary functions

2. Train doc2vec with a vector size of 300 and vectorize the text



```python
paragraphs = [item['paragraph'] for item in columns_paragraphs]

with open('../../Datasets/Text/preprocessed2.json', "r") as file:

# Preprocess the data
tagged_data = [TaggedDocument(doc, [i]) for i, doc in enumerate(preprocessed_paragraphs)]

# Train Doc2Vec model
max_epochs = 80
vec_size = 300
alpha = 0.025

model = Doc2Vec(vector_size=vec_size,
                alpha=alpha,
                min_alpha=0.05,
                min_count=1,
                dm =1)

model.build_vocab(tagged_data)

model.train(tagged_data,
            total_examples=model.corpus_count,
            epochs=model.epochs)
```

```python
vectors = [model.infer_vector(text) for text in preprocessed_paragraphs]
```

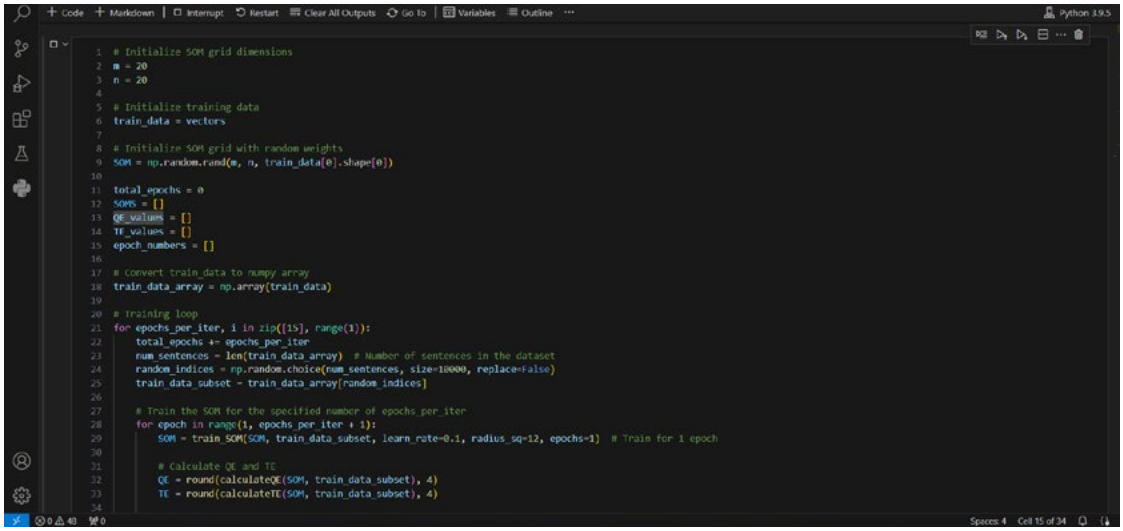
* Training Doc2Vec and vectorize the text

3. Define a function to fit the PCA on all cells of the SOM.

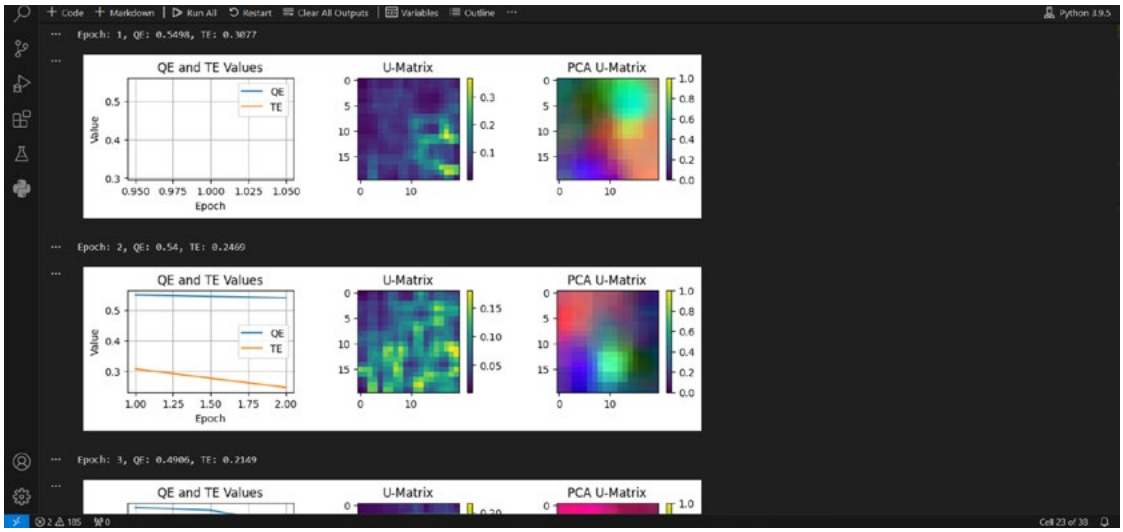
Fit PCA on all cells of the SOM

```python
def plot_som_pca(SOM):
    # Reshape the SOM grid into a 2D array
    som_flat = SOM.reshape(-1, SOM.shape[-1])

    # Perform PCA with 3 components
    pca = PCA(n_components=3)
    som_pca = pca.fit_transform(som_flat)

    # Normalize PCA components to [0, 1]
    som_pca_norm = (som_pca - som_pca.min(axis=0)) / (som_pca.max(axis=0) - som_pca.min(axis=0))

    # Create a U-matrix with PCA components as RGB values
    u_matrix_pca = som_pca_norm.reshape(SOM.shape[:-1] + (3,))

    # Plot the U-matrix with PCA colors
    # plt.figure(figsize=(4, 4))
    plt.imshow(u_matrix_pca)
    plt.title('PCA U-Matrix')
    plt.colorbar()
    plt.show()

def plot_epoch_results(epoch_number, QE_values, TE_values, SOM):
    # Plot QE and TE values
    plt.figure(figsize=(10, 4))
    plt.subplot(231)
    plt.plot(epoch_numbers, QE_values, label='QE')
    plt.plot(epoch_numbers, TE_values, label='TE')
    plt.xlabel('Epoch')
    plt.ylabel('Value')
    plt.title('QE and TE Values')
```

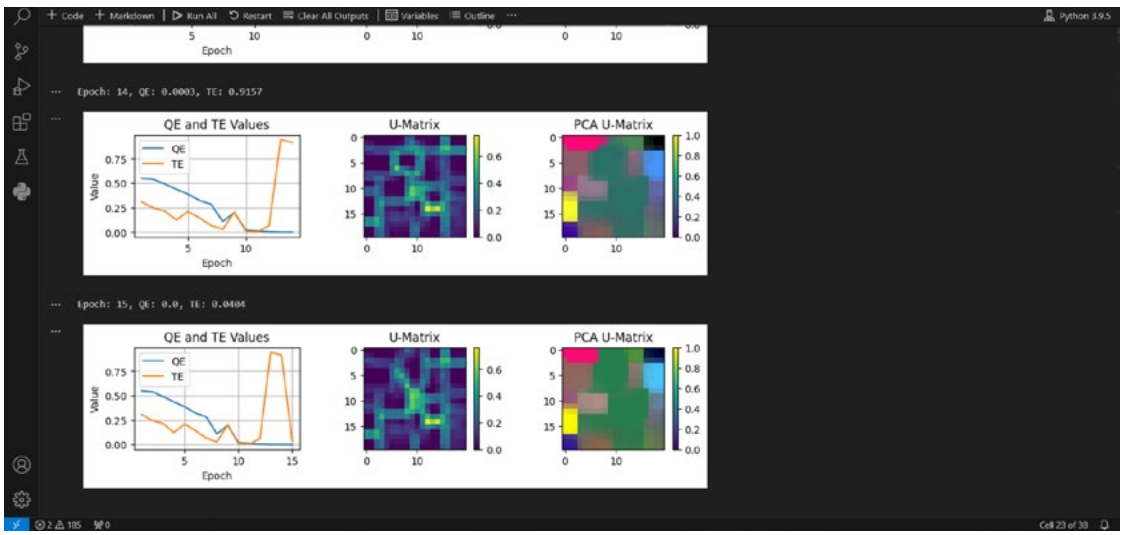
4. Set the size of the training data for each epoch to 10,000 (line 24) and number of training epochs (line 21). The training code saves the SOM, QE, and TE after each epoch each a list to be recalled later. While the code is running, it prints the epoch number, QE and TE values. Also, it plots a graphs of the QE, TE, U-Matrix, and PCA U-Matrix after each epoch.



```
# Initialize SOM grid dimensions
m = 20
n = 20

# Initialize training data
train_data = vectors

# Initialize SOM grid with random weights
SOM = np.random.rand(m, n, train_data[0].shape[0])

total_epochs = 0
SOMS = []
QE_values = []
TE_values = []
epoch_numbers = []

# Convert train_data to numpy array
train_data_array = np.array(train_data)

# Training loop
for epochs_per_iter, i in zip([15], range(1)):
    total_epochs += epochs_per_iter
    num_sentences = len(train_data_array)  # Number of sentences in the dataset
    random_indices = np.random.choice(num_sentences, size=10000, replace=False)
    train_data_subset = train_data_array[random_indices]

    # Train the SOM for the specified number of epochs_per_iter
    for epoch in range(1, epochs_per_iter + 1):
        SOM = train_SOM(SOM, train_data_subset, learn_rate=0.1, radius_sq=12, epochs=1)  # Train for 1 epoch

        # Calculate QE and TE
        QE = round(calculateQE(SOM, train_data_subset), 4)
        TE = round(calculateTE(SOM, train_data_subset), 4)
```

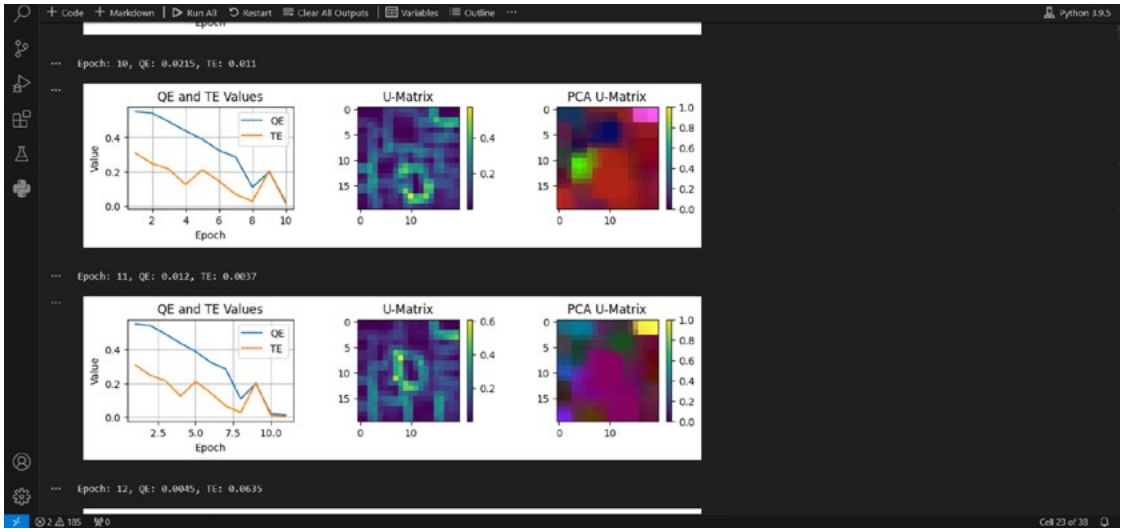


* Plots and prints after each epoch

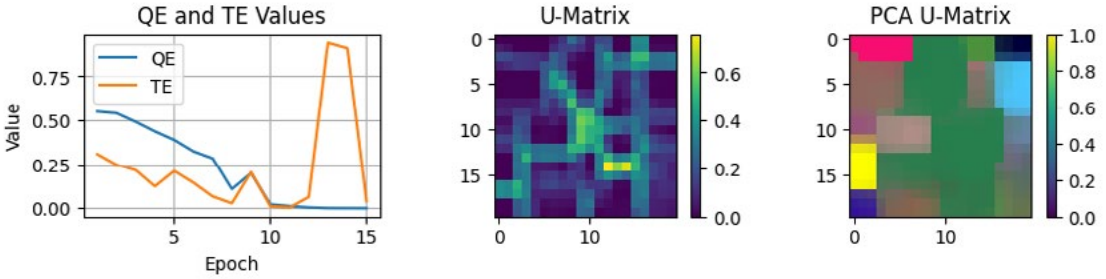


* Last epoch graph

As illustrated in the last printed graph, the QE and TE values were decaying till reaching epoch 11. Hence, epoch 10 is selected as it has the closest QE and TE values to zero but not zero. Altough epoch 11 has values closer to zero, however these values are almost zero which impacts the quality fo the SOM.



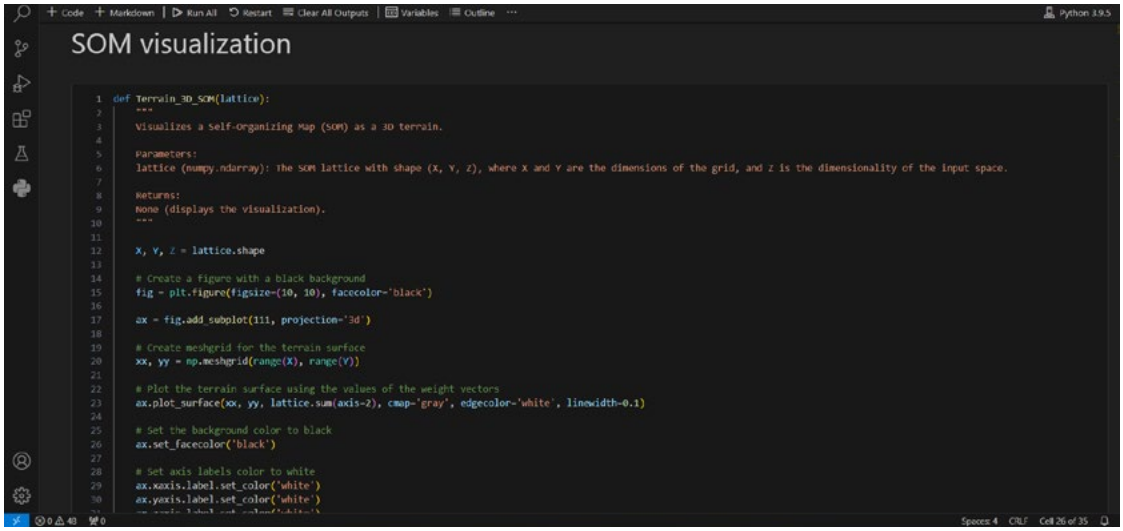
* Epoch 10 data



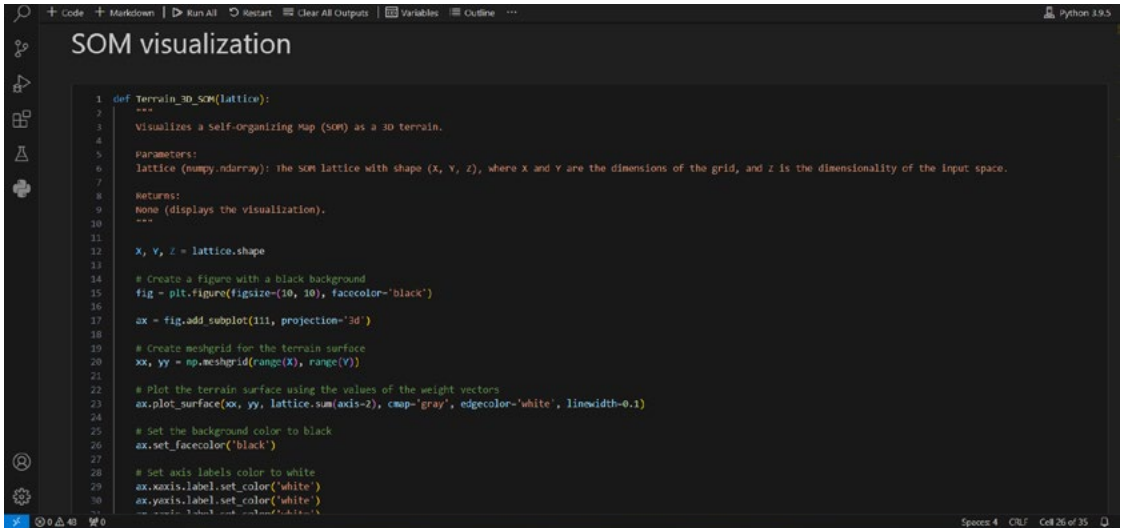
* Last epoch data graph

5. Print the length of the list “SOMS” and the content of “QE_values” and “TE_values” to ensure that the data of each epoch was saved during the training.

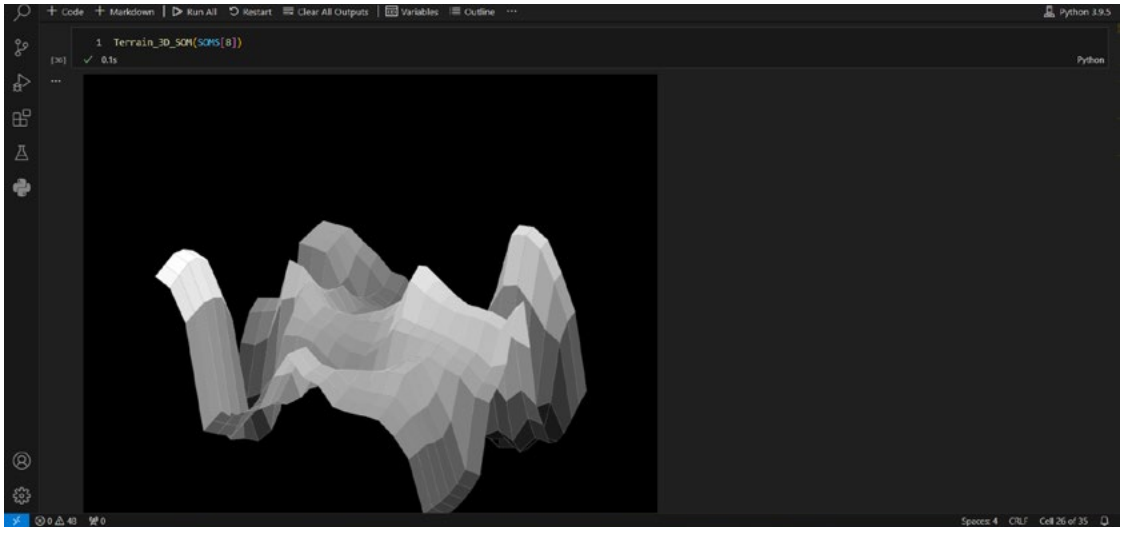
6. Define a function to fit PCA on the entire training set.

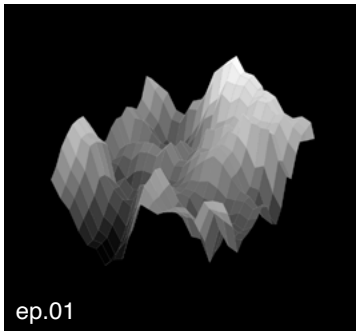


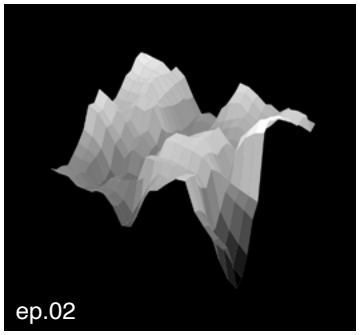
7. Define a function to plot the SOM as a 3D terrain

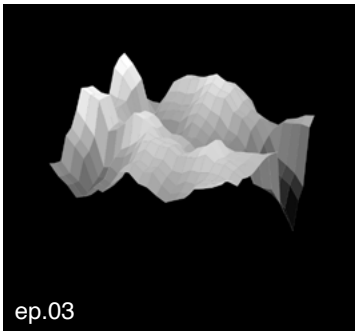


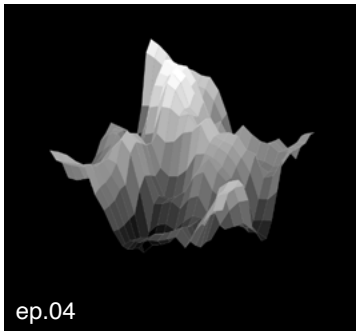
8. Plot the SOM as 3D terrain

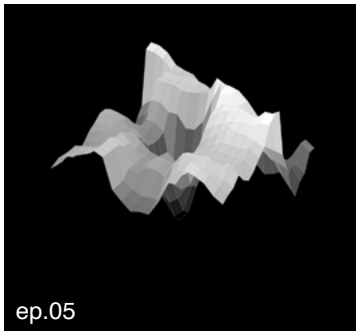


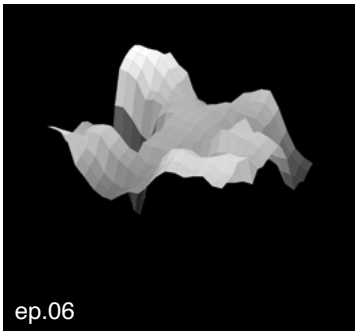


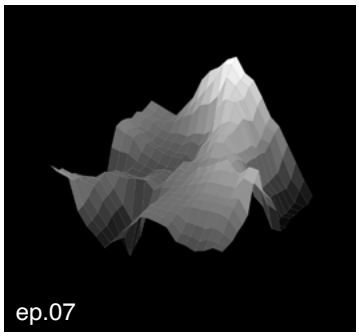


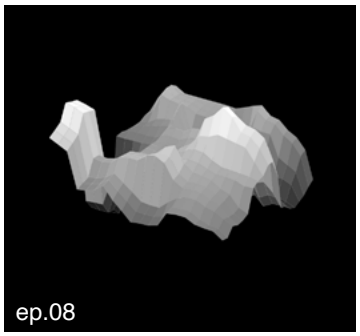


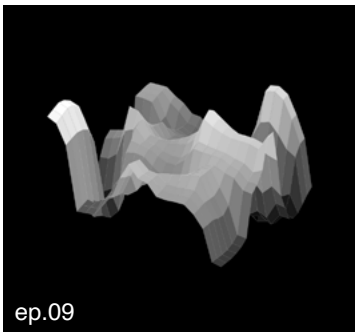


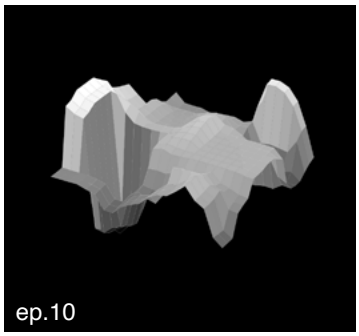


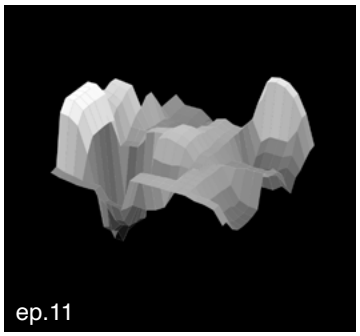


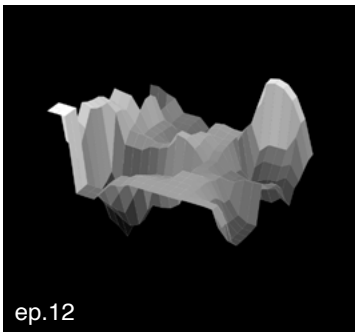


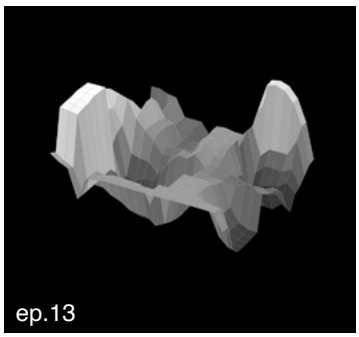


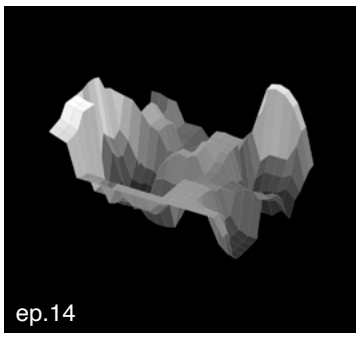


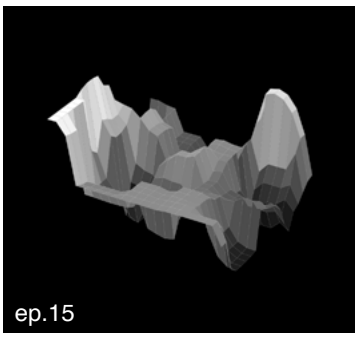






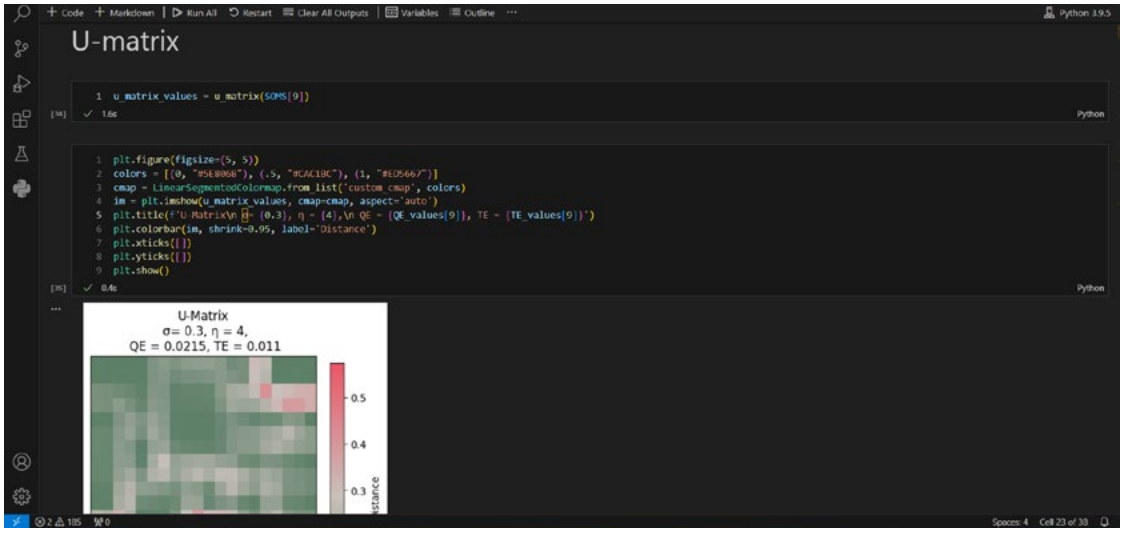


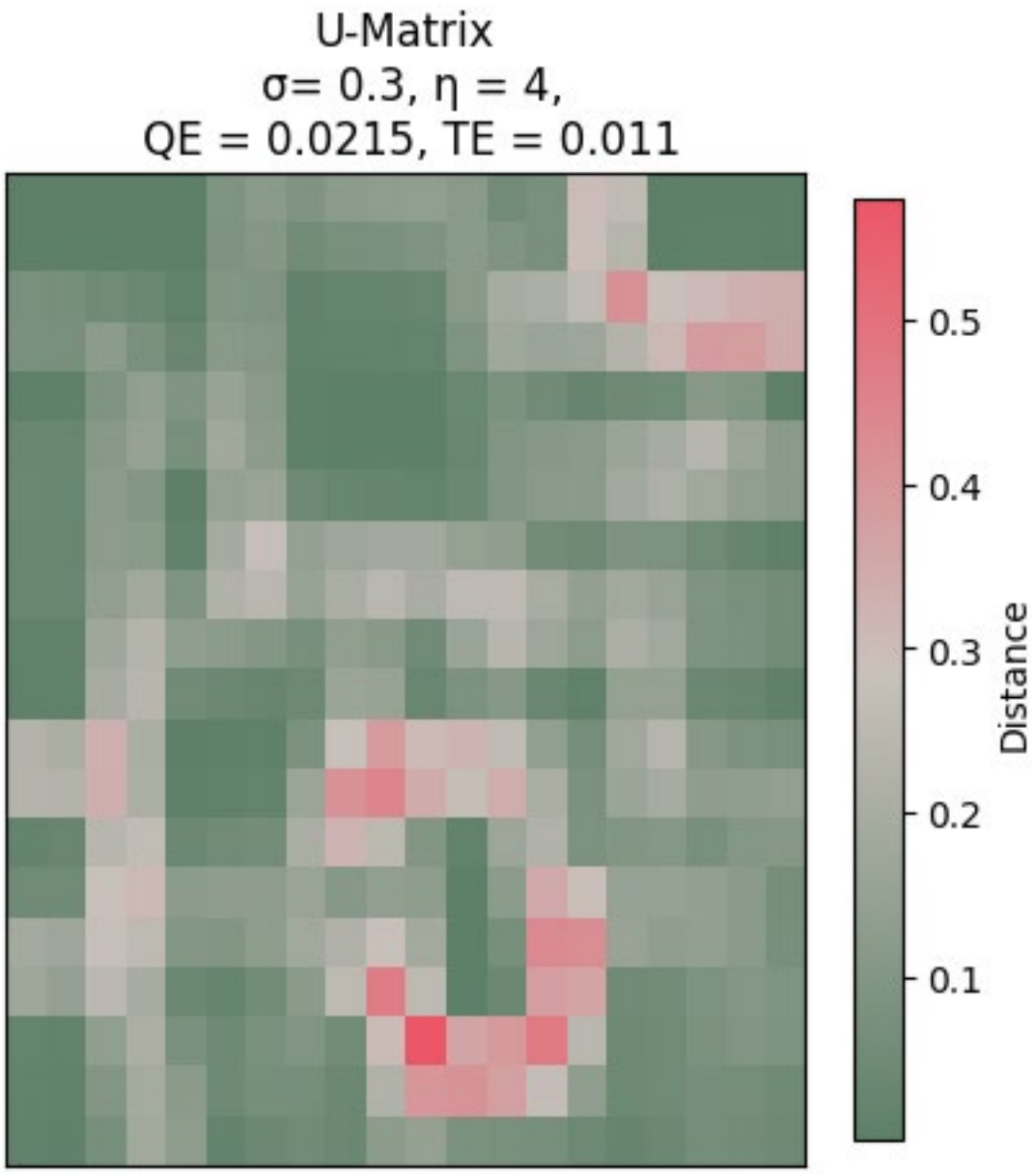




* One of the provocative comments we received during term 02 final crit is the uncovered spatial potential of SOMs. Based on that, I tried to translate the SOM into a form that can be translated physically. This does not mean this physical form can be turned into architecture. Rather, it has the potential to represent and visualize SOMs in physical models that can be 3D printed to explore a new attribute of SOMs further than 2D representations.

9. Plot the U-Matrix of the SOM at epoch 10



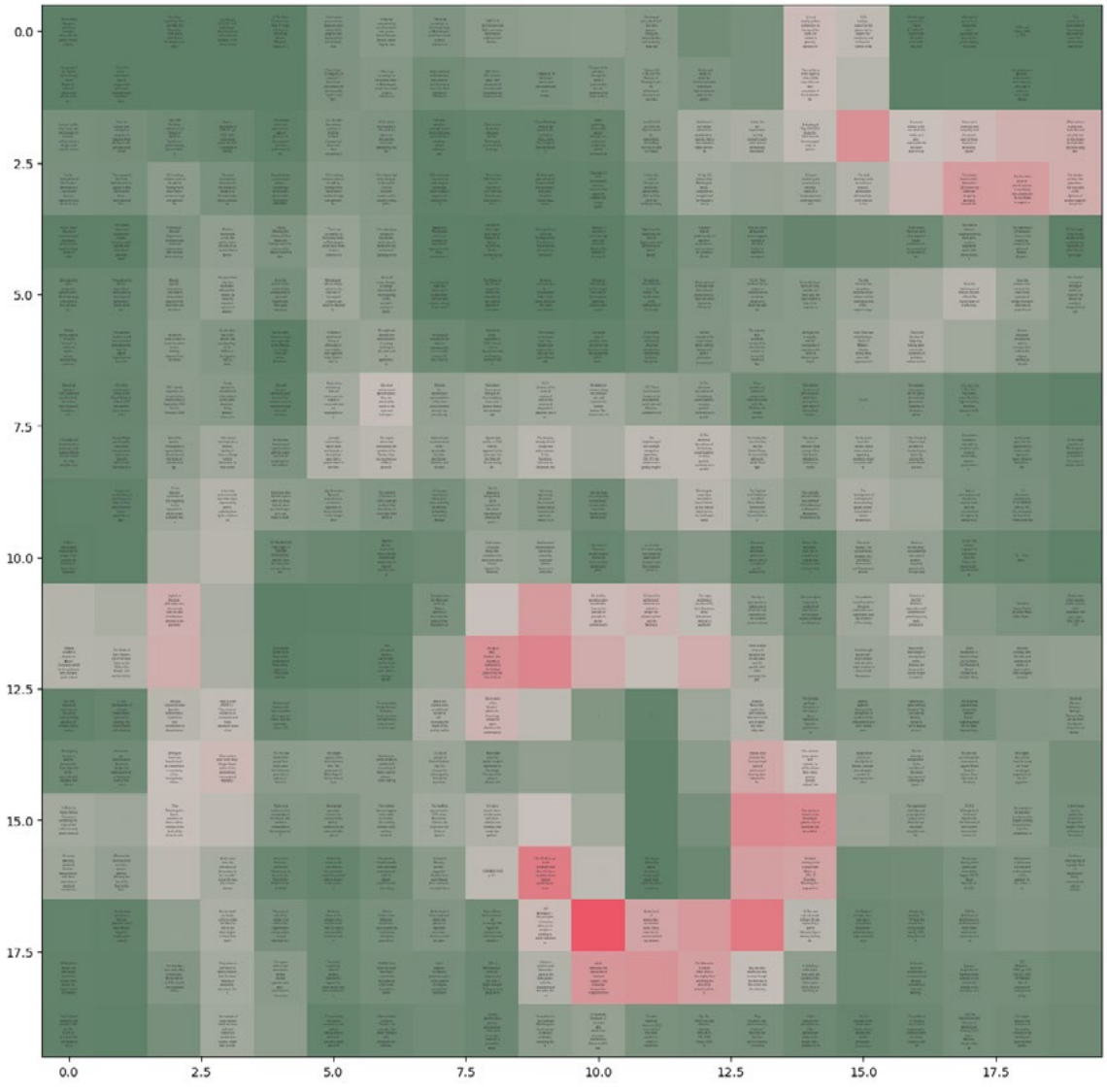


10. Print the sentences of each cell on the U-Matrix

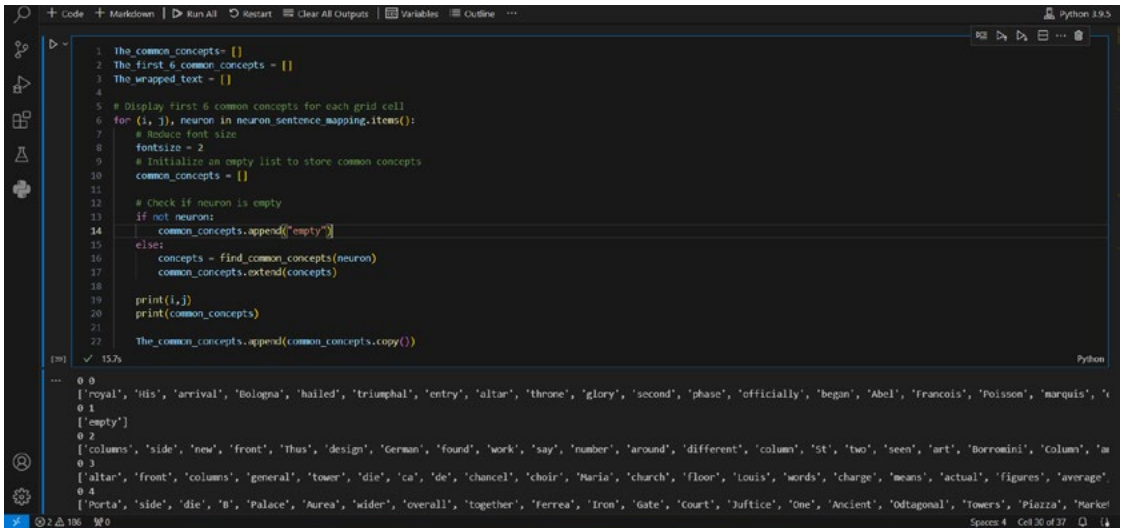
```
# Initialize an empty dictionary to store sentences associated with each neuron
neuron_sentence_mapping = {(i, j): [] for i in range(SOM.shape[0]) for j in range(SOM.shape[1])}
```

```
# Iterate through each sentence and find its BMU, then append it to the corresponding neuron's list
for idx, sentence in enumerate(paragraphs):
    vector = vectors[idx]
    bmu = find_BMU(SOMS[9], vector)
    neuron_sentence_mapping[bmu].append(sentence)
```

```
import textwrap

# Plot the U-Matrix and display the mapped sentences for each neuron
plt.figure(figsize=(15, 15))
im = plt.imshow(u_matrix_values, cmap=cmap, aspect='auto')
# plt.title(f'U-Matrix\n σ= {0.3}, η = {4},\n QE = {QE}, TE = {TE}')
# plt.colorbar(im, orientation='horizontal', label='Distance', pad=0.01)

# Display only the first 100 characters of each sentence in each grid cell
for (i, j), sentences in neuron_sentence_mapping.items():
    if sentences:  # check if there are sentences associated with the neuron
        # Reduce font size
        fontsize = 2
        # Select the first sentence associated with the neuron
        sentence = sentences[0][:100]  # Take the first 100 characters
        # Wrap the sentence to fit within the grid cell
        wrapped_sentence = "\n".join(textwrap.wrap(sentence, width=15))
        plt.text(j, i, wrapped_sentence, ha='center', va='center', color='#353637', fontsize=fontsize)

plt.savefig('SOM_9.pdf', format='pdf')
```



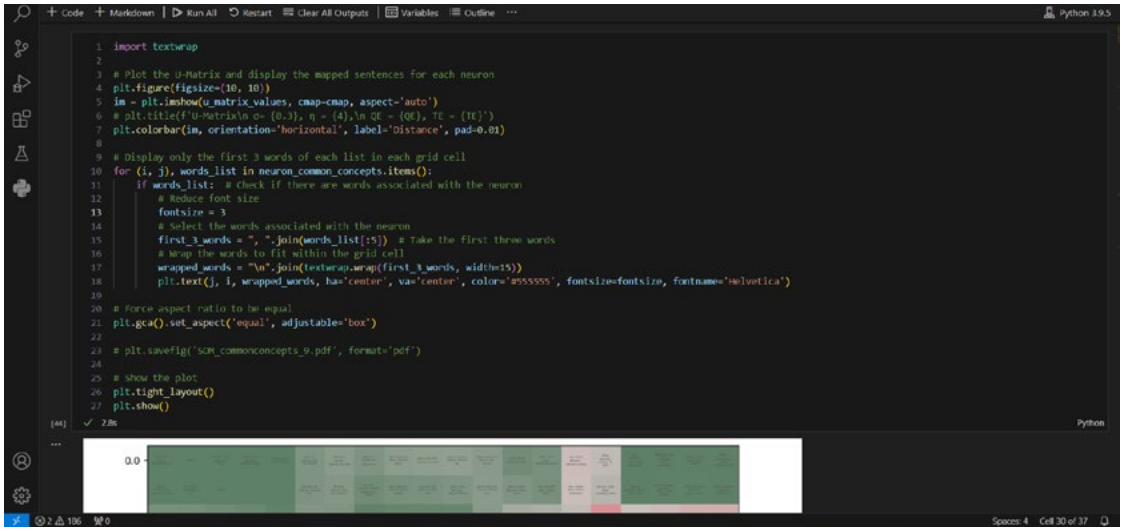
11. Define a function to find the common concepts of sentences in each cell



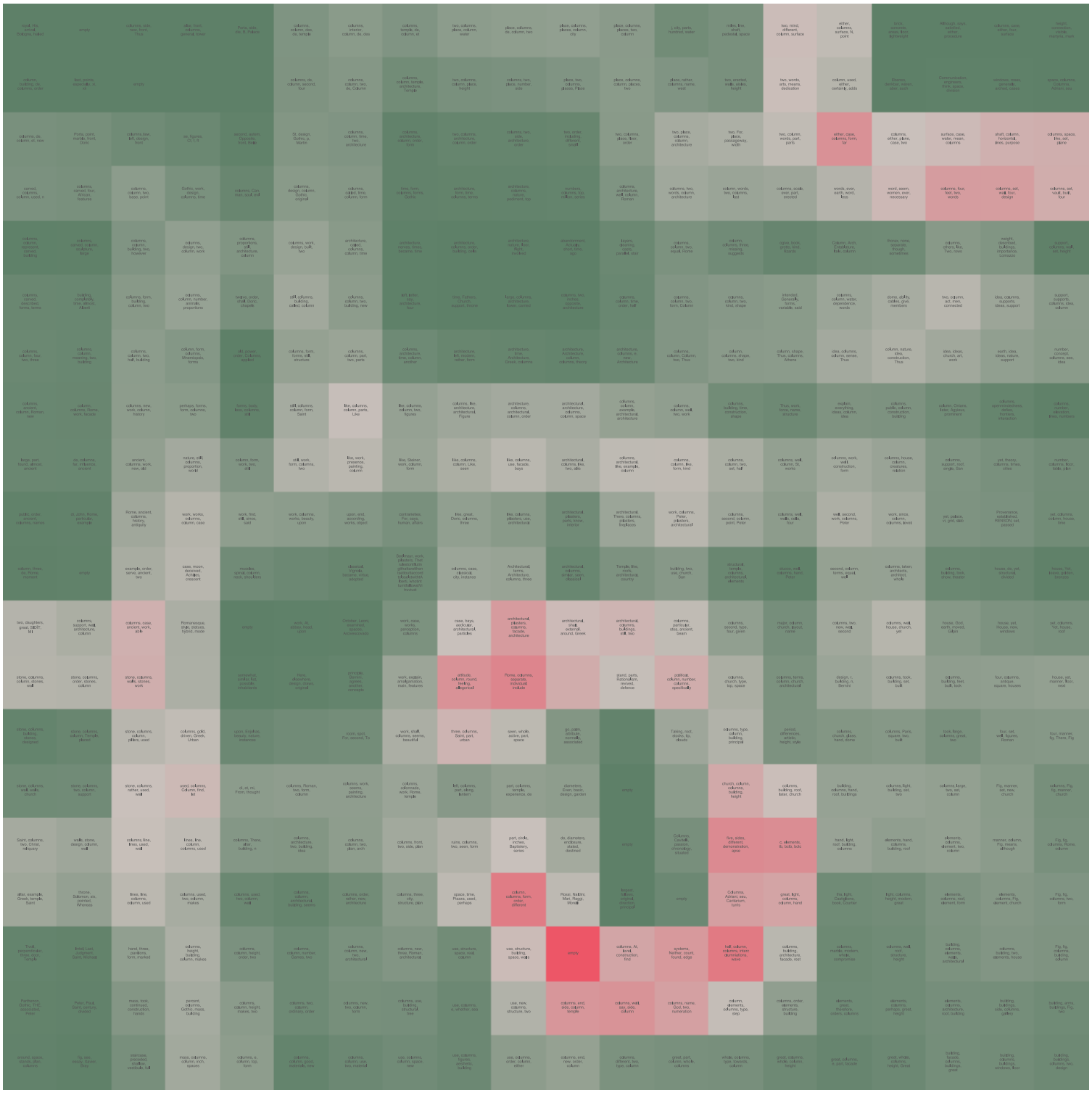
```
The_common_concepts= []
The_first_6_common_concepts = []
The_wrapped_text = []

# Display first 6 common concepts for each grid cell
for (i, j), neuron in neuron_sentence_mapping.items():
    # Reduce font size
    fontsize = 2
    # Initialize an empty list to store common concepts
    common_concepts = []

    # Check if neuron is empty
    if not neuron:
        common_concepts.append("empty")
    else:
        concepts = find_common_concepts(neuron)
        common_concepts.extend(concepts)

    print(i,j)
    print(common_concepts)

    The_common_concepts.append(common_concepts.copy())
```

```
0 0
['royal', 'His', 'arrival', 'Bologna', 'hailed', 'triumphal', 'entry', 'altar', 'throne', 'glory', 'second', 'phase', 'officially', 'began', 'Abel', 'Francois', 'Poisson', 'marquis',
0 1
['empty']
0 2
['columns', 'side', 'new', 'front', 'Thus', 'design', 'German', 'found', 'work', 'say', 'number', 'around', 'different', 'column', 'St', 'two', 'seen', 'art', 'Borromini', 'Column',
0 3
['altar', 'front', 'columns', 'general', 'tower', 'die', 'ca', 'de', 'chancel', 'choir', 'Maria', 'church', 'floor', 'Louis', 'words', 'charge', 'means', 'actual', 'figures', 'average'
0 4
['Porta', 'side', 'die', 'B', 'Palace', 'Aurea', 'wider', 'overall', 'together', 'Ferrea', 'Iron', 'Gate', 'Court', 'Juftice', 'One', 'Ancient', 'Odtagonal', 'Towers', 'Piazza', 'Market
```

12. Print the common concept of each cell on the SOM

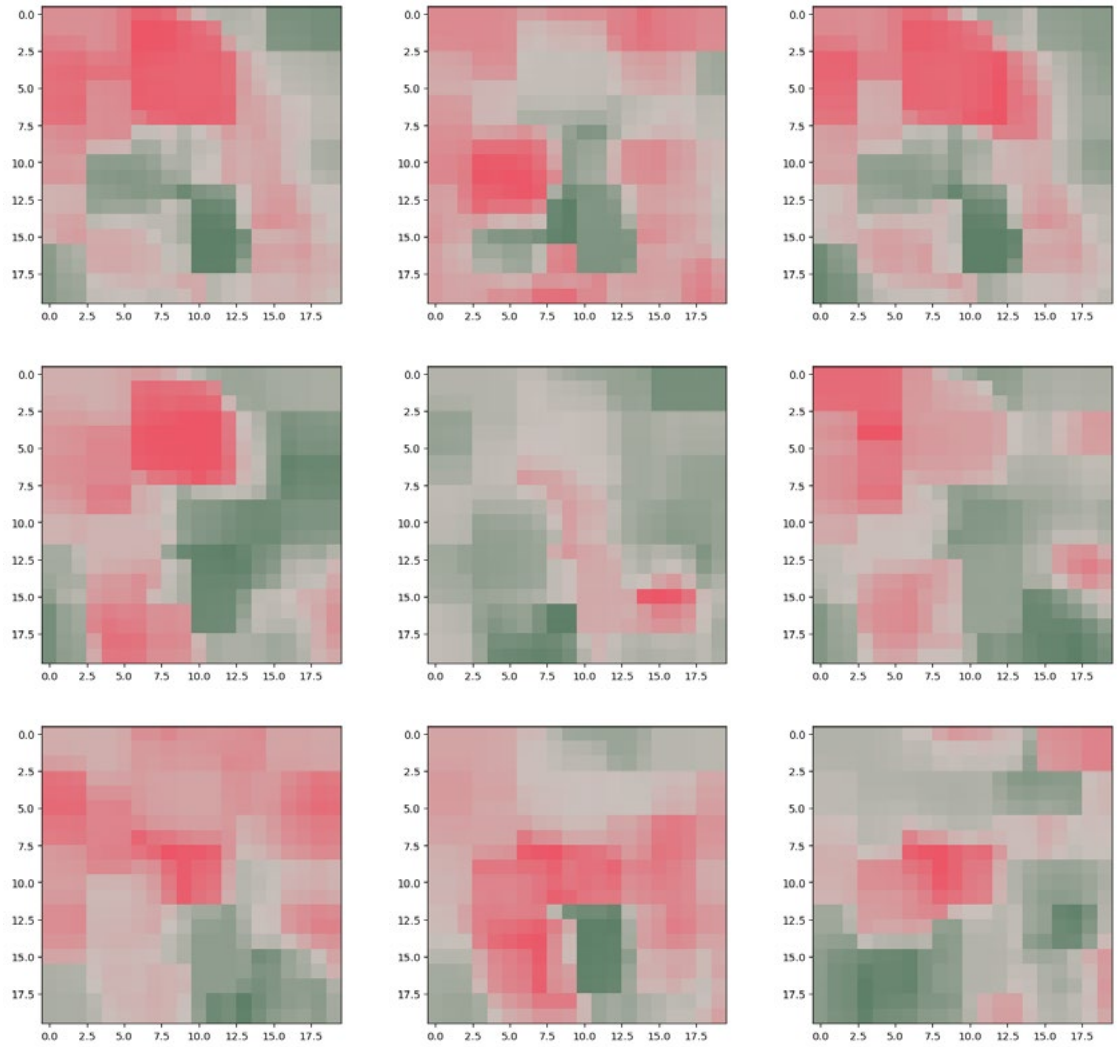


```
import textwrap

# Plot the U-Matrix and display the mapped sentences for each neuron
plt.figure(figsize=(10, 10))
im = plt.imshow(u_matrix_values, cmap=cmap, aspect='auto')
# plt.title(f'U-Matrix\n σ= {σ:.3}, η = {η},\n QE = {QE}, TE = {TE}')
plt.colorbar(im, orientation='horizontal', label='Distance', pad=0.01)

# Display only the first 3 words of each list in each grid cell
for (i, j), words_list in neuron_common_concepts.items():
    if words_list:  # Check if there are words associated with the neuron
        # Reduce font size
        fontsize = 3
        # Select the words associated with the neuron
        first_3_words = ", ".join(words_list[:5])  # Take the first three words
        # Wrap the words to fit within the grid cell
        wrapped_words = "\n".join(textwrap.wrap(first_3_words, width=15))
        plt.text(j, i, wrapped_words, ha='center', va='center', color='#555555', fontsize=fontsize, fontname='Helvetica')

# Force aspect ratio to be equal
plt.gca().set_aspect('equal', adjustable='box')

# plt.savefig('SOM_commonconcepts_9.pdf', format='pdf')

# Show the plot
plt.tight_layout()
plt.show()
```



13. Activate the SOM

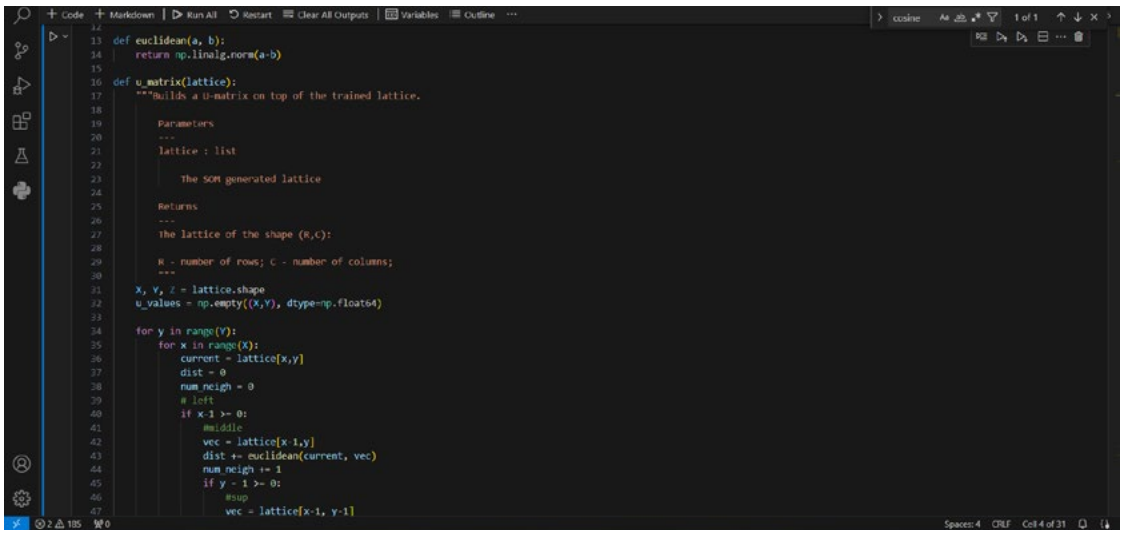
```python
# 01

# Step 1: Convert the query sentences into vectors
query_sentences = ["ionic column"]
query_vectors = [model.infer_vector(word_tokenize(sentence.lower())) for sentence in query_sentences]


# Step 2: Activate the SOM using the activate function for each query vector
activated_SOMs = []
for query_vector in query_vectors:
    activated_SOM = activate(train_data, SOMS[9], query_vector)
    activated_SOMs.append(activated_SOM)

# Step 2: Activate the SOM using the activate function
activated_SOM = activate(train_data, SOMS[9], query_vector)

# plot
plt.figure(figsize=(5, 5))
im = plt.imshow(activated_SOM, cmap=cmap, aspect='auto')

plt.show()
```



* SOM activations

IMAGES

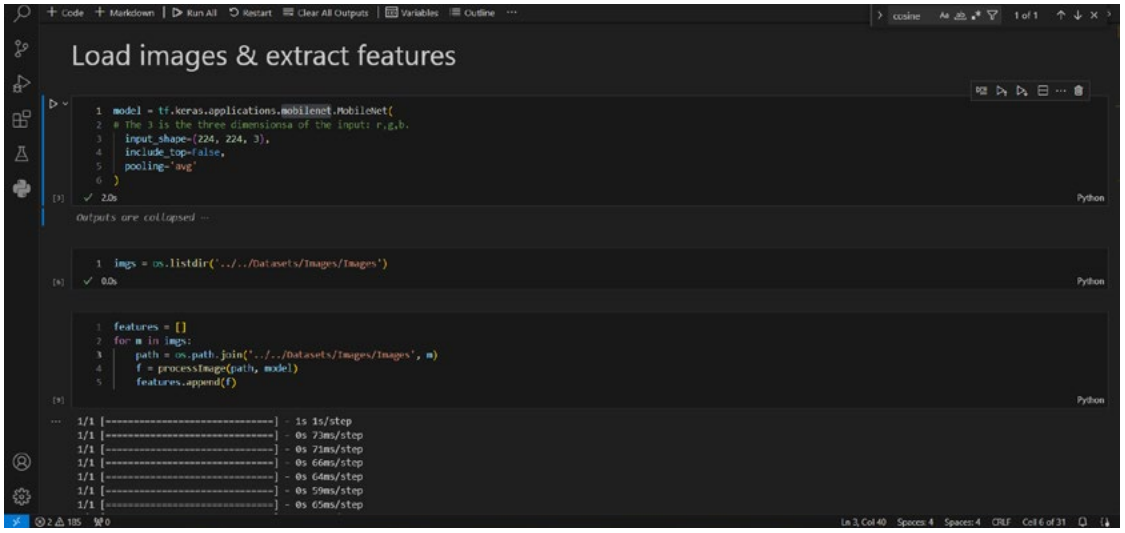
1. Define the functions, with processing images Euclidean distance is used instead of cosine_similarity.



```
def euclidean(a, b):
    return np.linalg.norm(a-b)

def u_matrix(lattice):
    """Builds a U-matrix on top of the trained lattice.

        Parameters
        ---
        lattice : list

            The SOM generated lattice

        Returns
        ---
        The lattice of the shape (R,C):

        R - number of rows; C - number of columns;
        """
    X, Y, Z = lattice.shape
    u_values = np.empty((X,Y), dtype=np.float64)

    for y in range(Y):
        for x in range(X):
            current = lattice[x,y]
            dist = 0
            num_neigh = 0
            # left
            if x-1 >= 0:
                #middle
                vec = lattice[x-1,y]
                dist += euclidean(current, vec)
                num_neigh += 1
                if y - 1 >= 0:
                    #sup
                    vec = lattice[x-1, y-1]
```

* Defining necessary functions

2. Load mobilenet as a model and extract the features of the 4411 images



Load images & extract features

```
model = tf.keras.applications.mobilenet.MobileNet(
# The 3 is the three dimensionsa of the input: r,g,b.
  input_shape=(224, 224, 3),
  include_top=false,
  pooling='avg'
)
```

2.0s Python

Outputs are collapsed ...

```
imgs = os.listdir('../../Datasets/Images/Images')
```

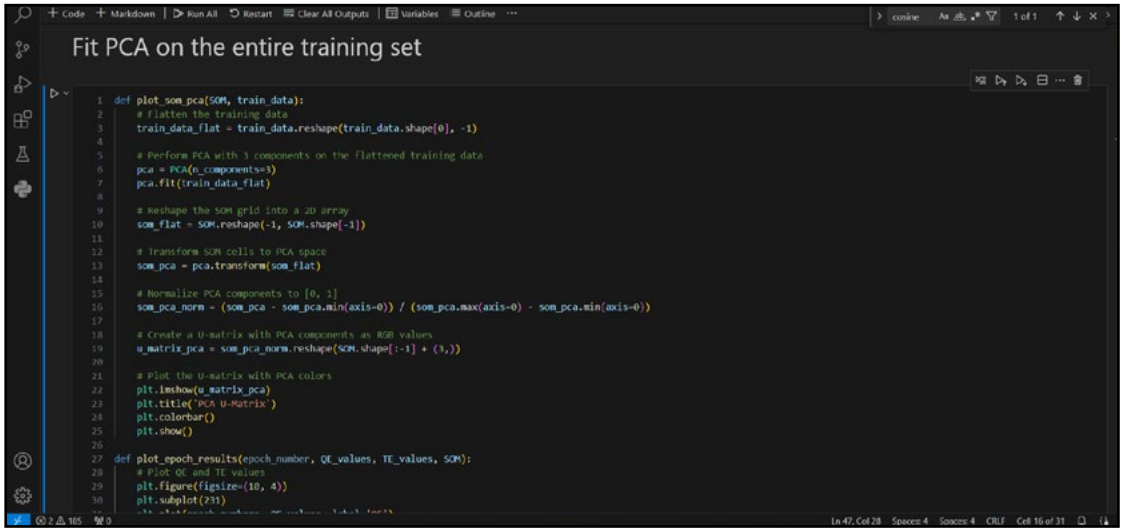
0.0s Python

```
features = []
for m in imgs:
    path = os.path.join('../../Datasets/Images/Images', m)
    f = processImage(path, model)
    features.append(f)
```

Python

```
1/1 [==============================] - 1s 1s/step
1/1 [==============================] - 0s 73ms/step
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 65ms/step
```

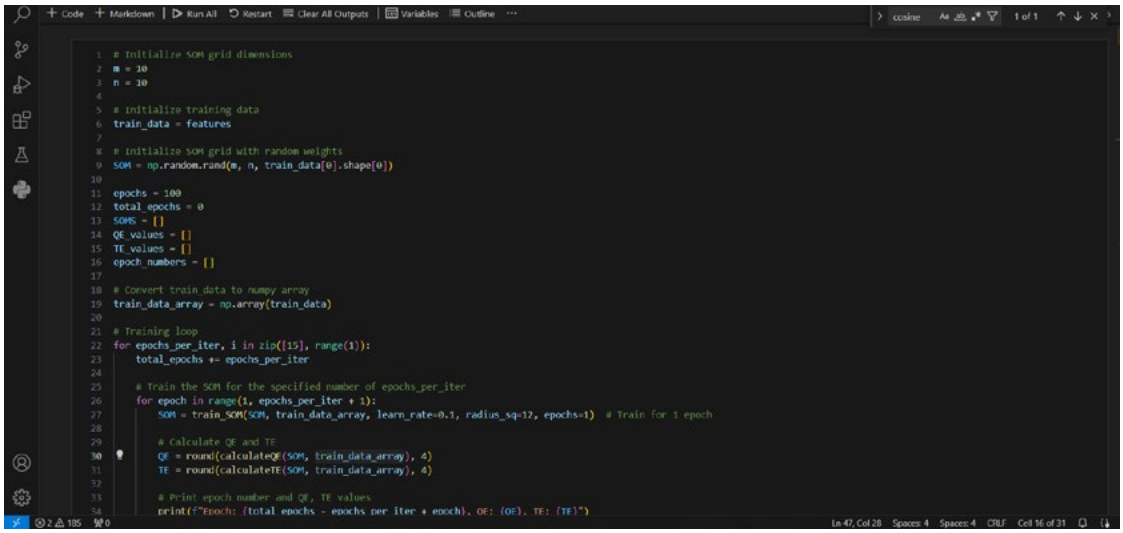
3. Define a function to fit the PCA on all cells of the SOM.



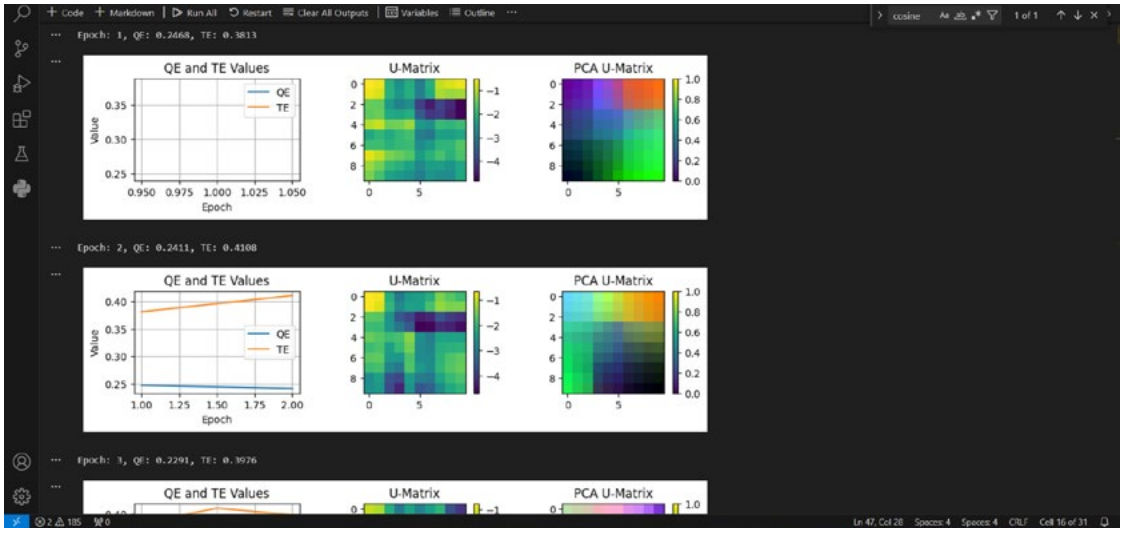
Fit PCA on the entire training set

```
def plot_som_pca(SOM, train_data):
    # Flatten the training data
    train_data_flat = train_data.reshape(train_data.shape[0], -1)

    # Perform PCA with 3 components on the flattened training data
    pca = PCA(n_components=3)
    pca.fit(train_data_flat)

    # Reshape the SOM grid into a 2D array
    som_flat = SOM.reshape(-1, SOM.shape[-1])

    # Transform SOM cells to PCA space
    som_pca = pca.transform(som_flat)

    # Normalize PCA components to [0, 1]
    som_pca_norm = (som_pca - som_pca.min(axis=0)) / (som_pca.max(axis=0) - som_pca.min(axis=0))

    # Create a U-matrix with PCA components as RGB values
    u_matrix_pca = som_pca_norm.reshape(SOM.shape[:-1] + (3,))

    # Plot the U-matrix with PCA colors
    plt.imshow(u_matrix_pca)
    plt.title('PCA U-Matrix')
    plt.colorbar()
    plt.show()

def plot_epoch_results(epoch_number, QE_values, TE_values, SOM):
    # Plot QE and TE values
    plt.figure(figsize=(10, 4))
    plt.subplot(231)
```

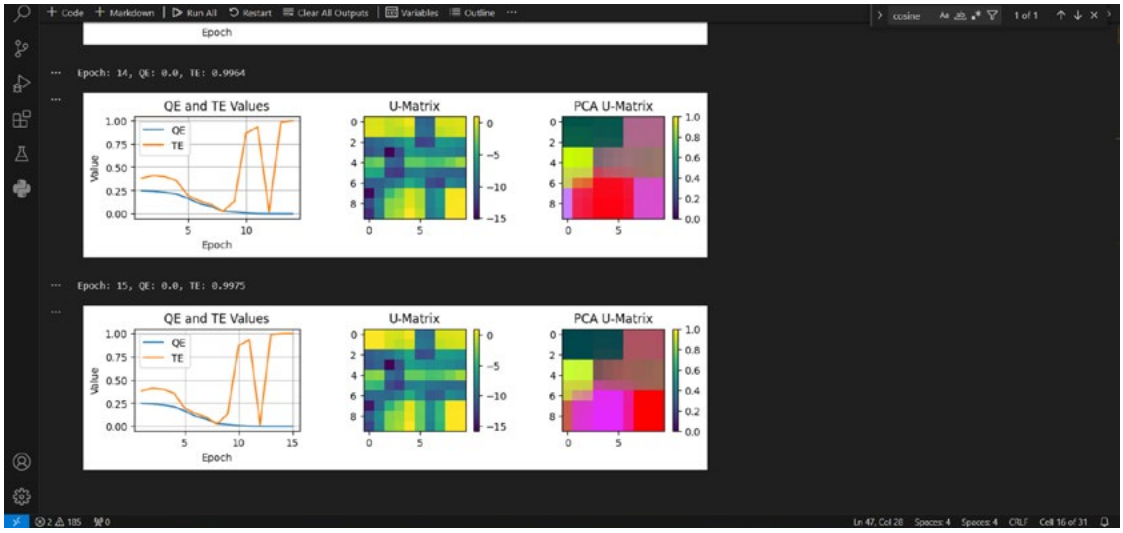
4. Train a 10x10 SOM using the extracted features for 15 epochs.



```
# Initialize SOM grid dimensions
m = 10
n = 10

# initialize training data
train_data = features

# initialize SOM grid with random weights
SOM = np.random.rand(m, n, train_data[0].shape[0])

epochs = 100
total_epochs = 0
SOMS = []
QE_values = []
TE_values = []
epoch_numbers = []

# Convert train data to numpy array
train_data_array = np.array(train_data)

# Training loop
for epochs_per_iter, i in zip([15], range(1)):
    total_epochs += epochs_per_iter

    # Train the SOM for the specified number of epochs_per_iter
    for epoch in range(1, epochs_per_iter + 1):
        SOM = train_SOM(SOM, train_data_array, learn_rate=0.1, radius_sq=12, epochs=1)  # Train for 1 epoch

        # Calculate QE and TE
        QE = round(calculateQE(SOM, train_data_array), 4)
        TE = round(calculateTE(SOM, train_data_array), 4)

        # Print epoch number and QE, TE values
        print(f"Epoch: {total_epochs - epochs_per_iter + epoch}, QE: {QE}, TE: {TE}")
```

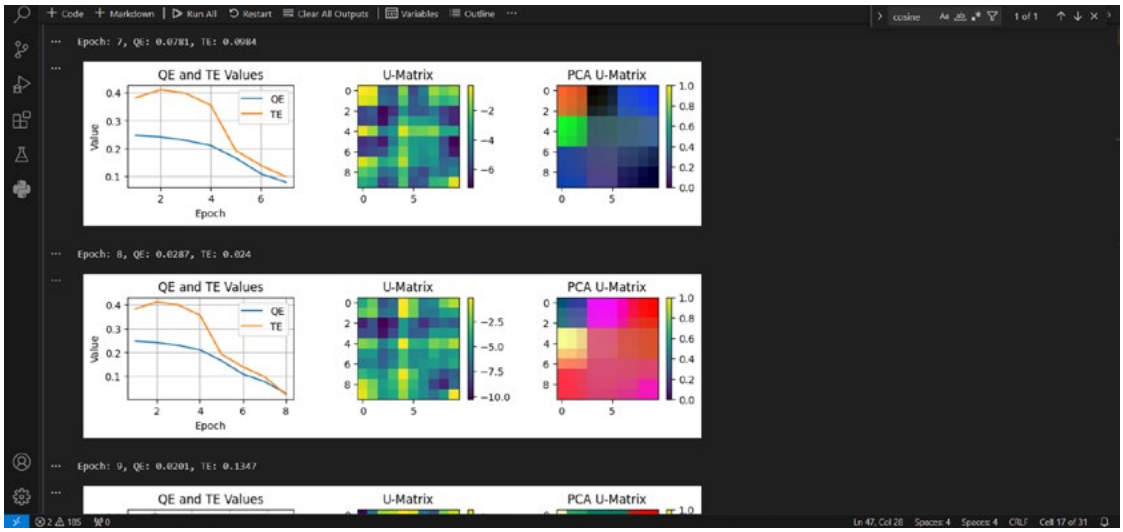


* Plots and prints after each epoch

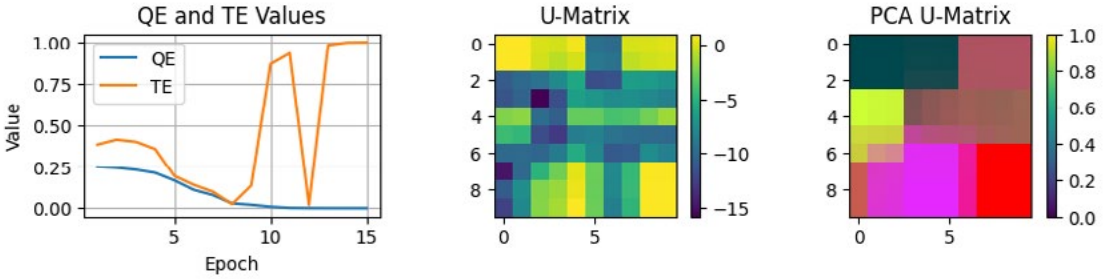


* Last epoch graph

5. Epoch 8 is selected as it has the QE and TE values close to zero. Also, it is noteworthy to mention that the QE and TE values spiked after this epoch as illustrated by the graph.

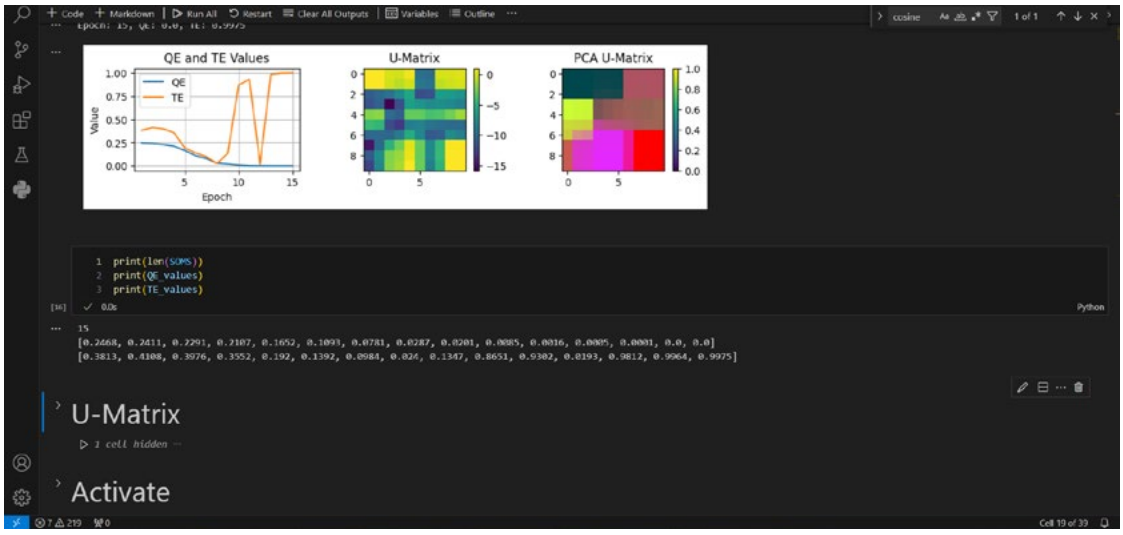


* Epoch 8 data

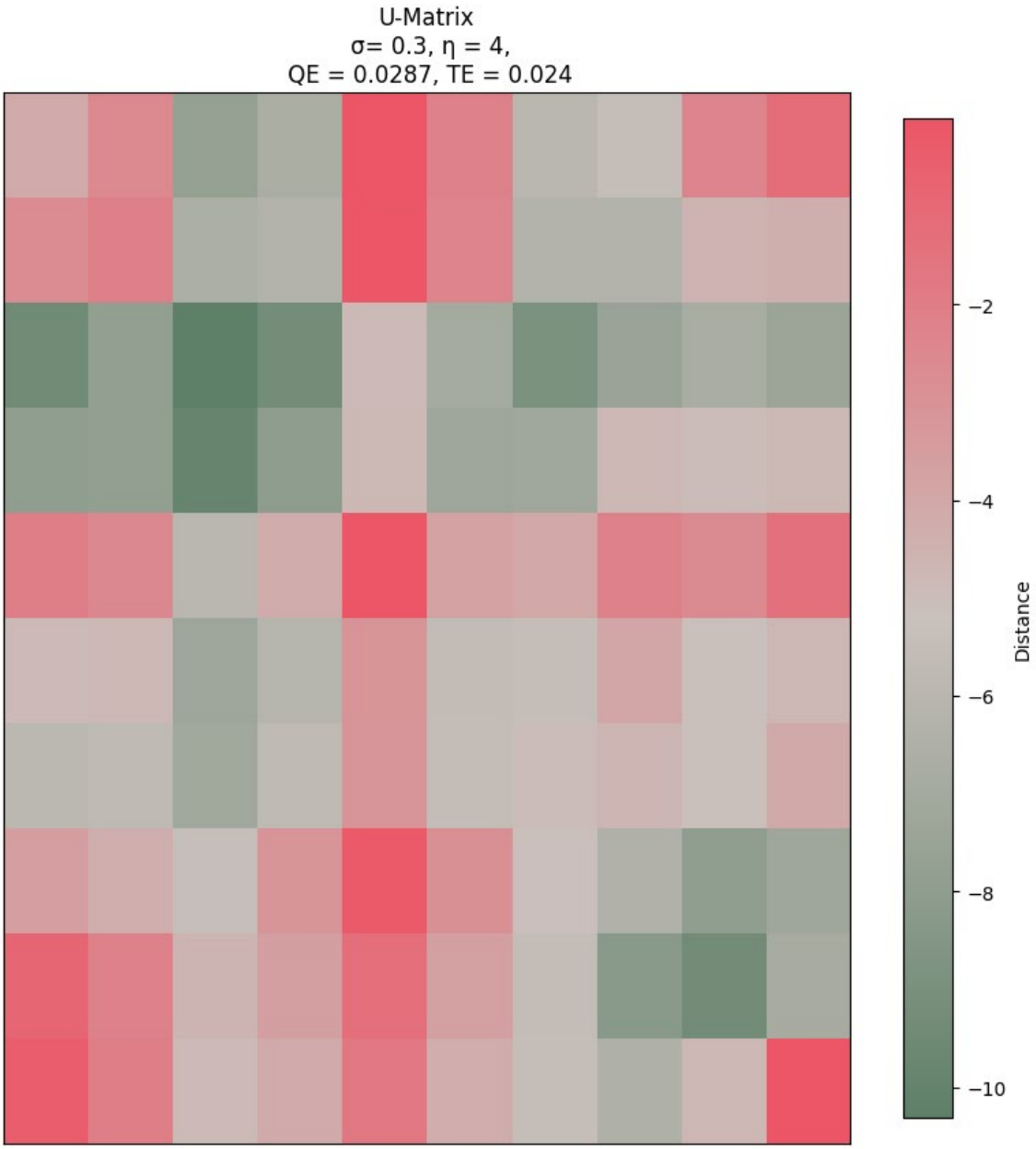


* Last epoch data graph

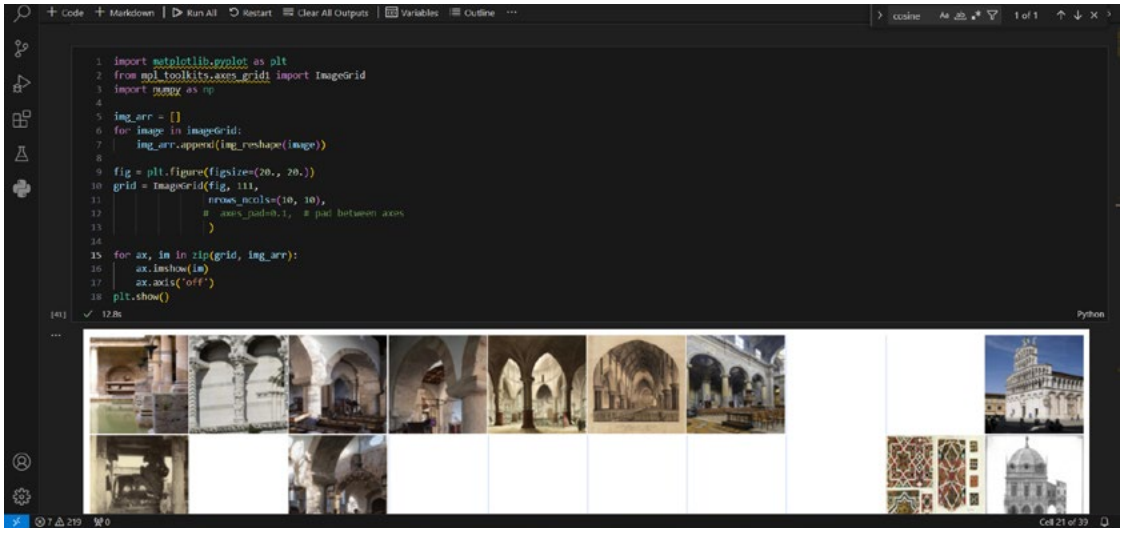
6. Print the length of the list “SOMS” and the content of “QE_values” and “TE_values” to ensure that the data of each epoch was saved during the training.



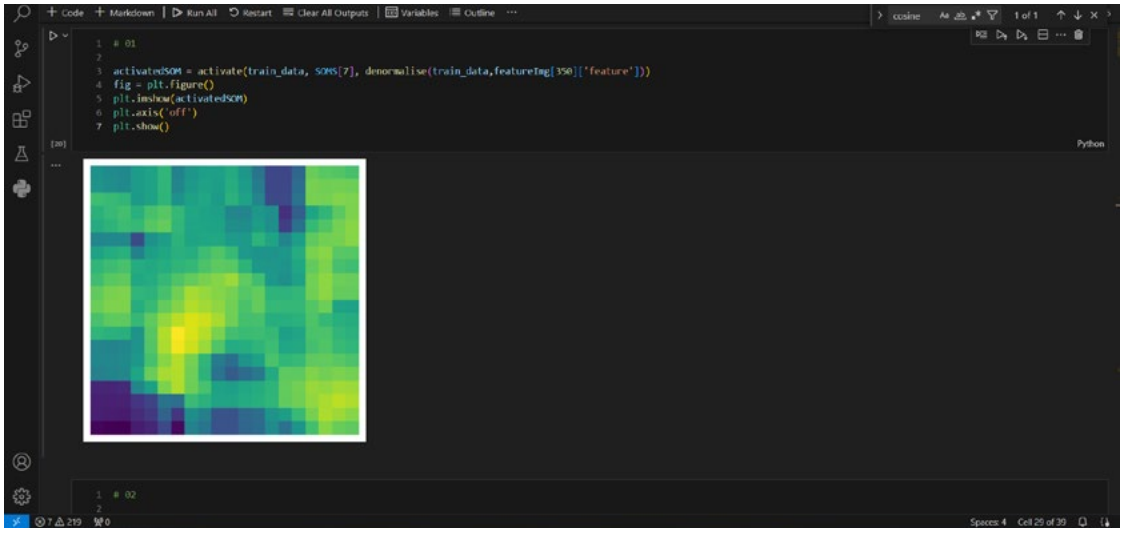
7. Plot the U-Matrix of the SOM at epoch 8

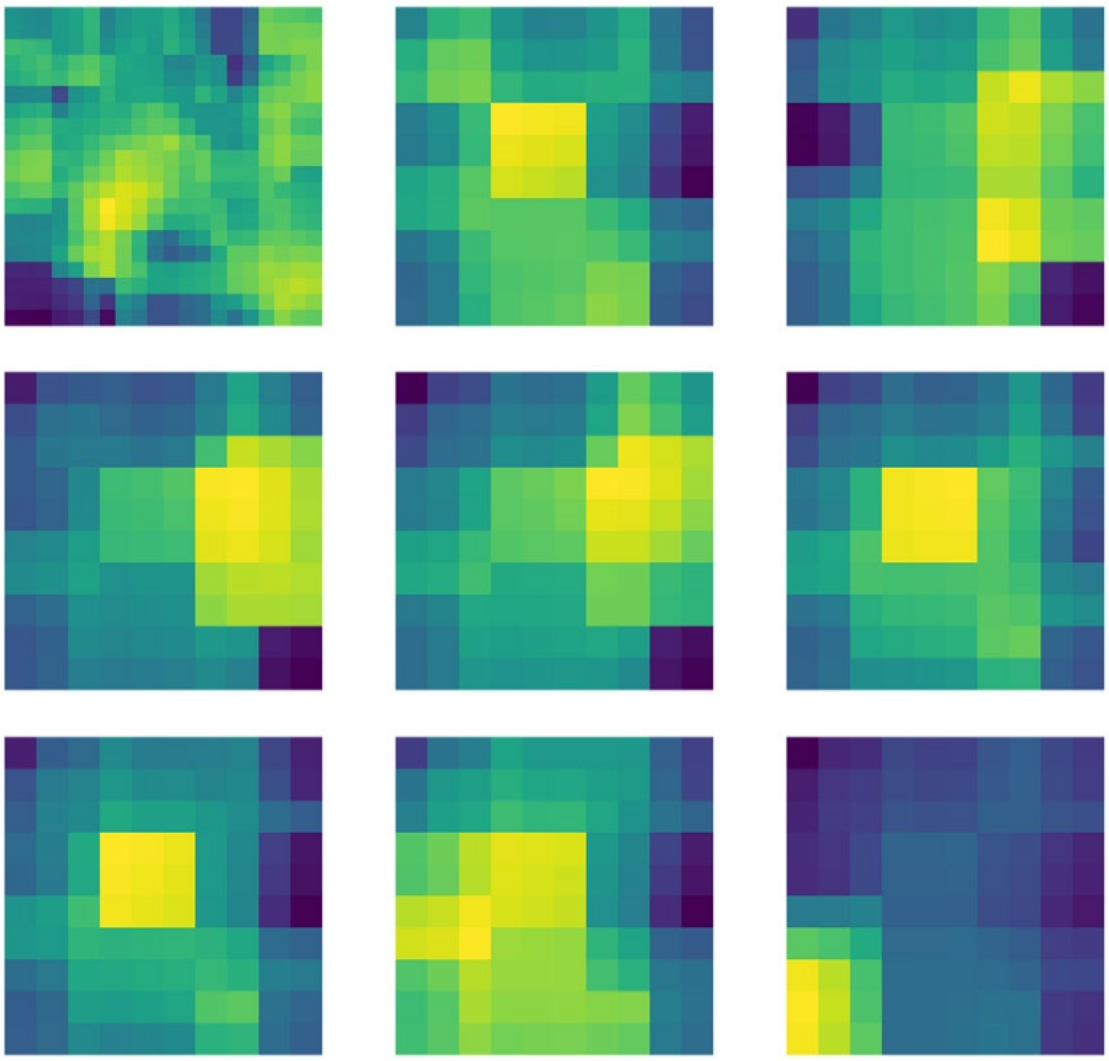


8. Plot the images on the SOM



9. Activate the SOM

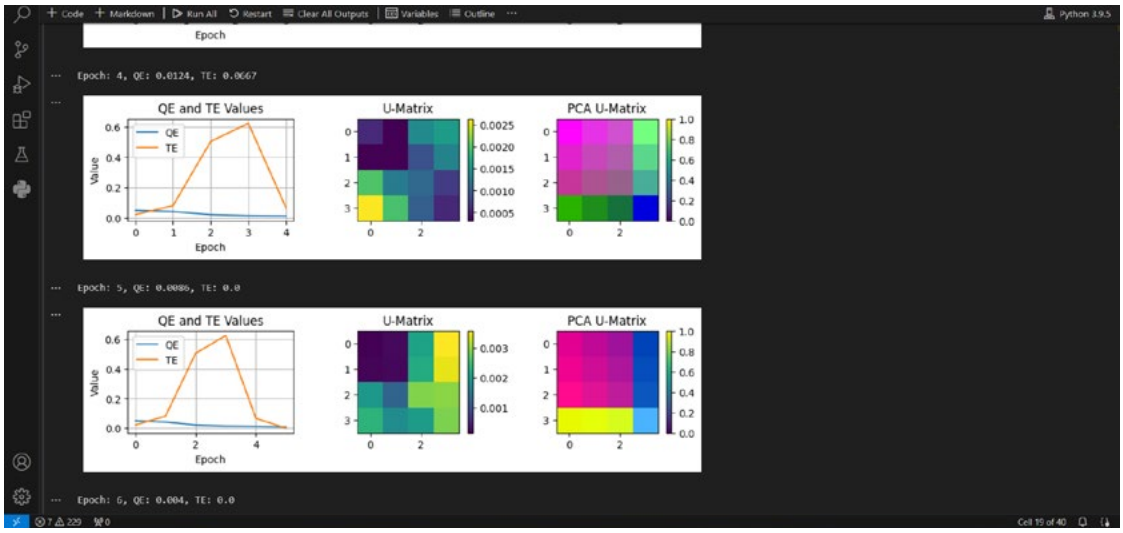




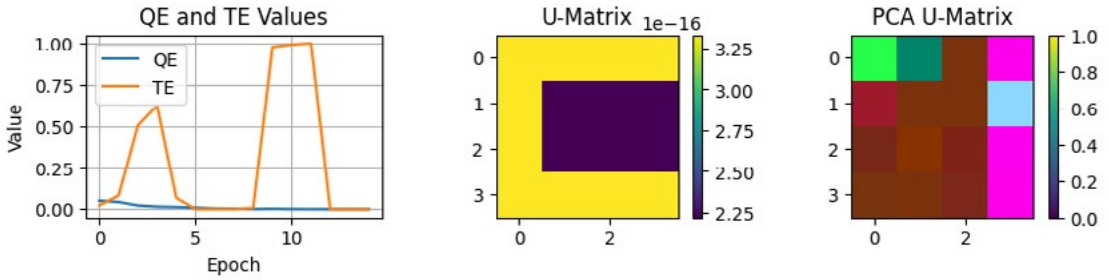
* SOM activations

VIDEOS

1. Proceed with the same process explained earlier. After training the SOM, analyze the data and select the best epoch. In this case, epoch 4 is selected as it has the closest QE and TE values to zero.

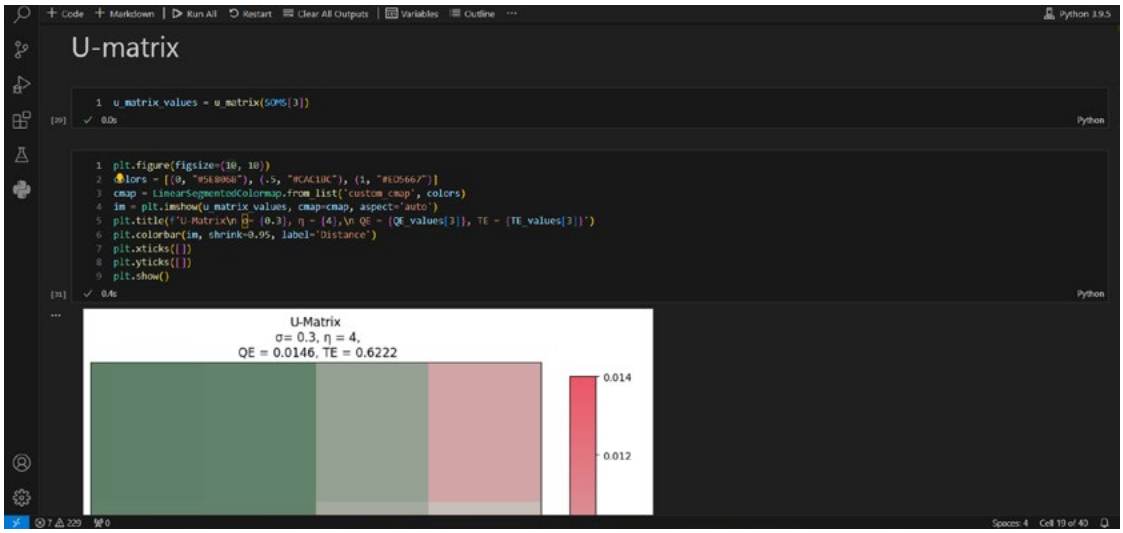


* Epoch 4 data

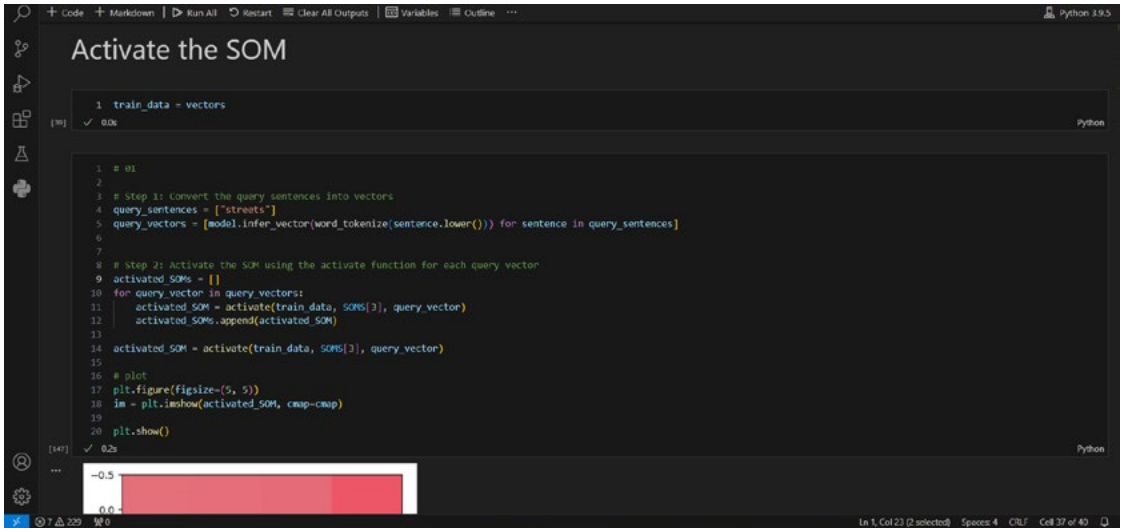


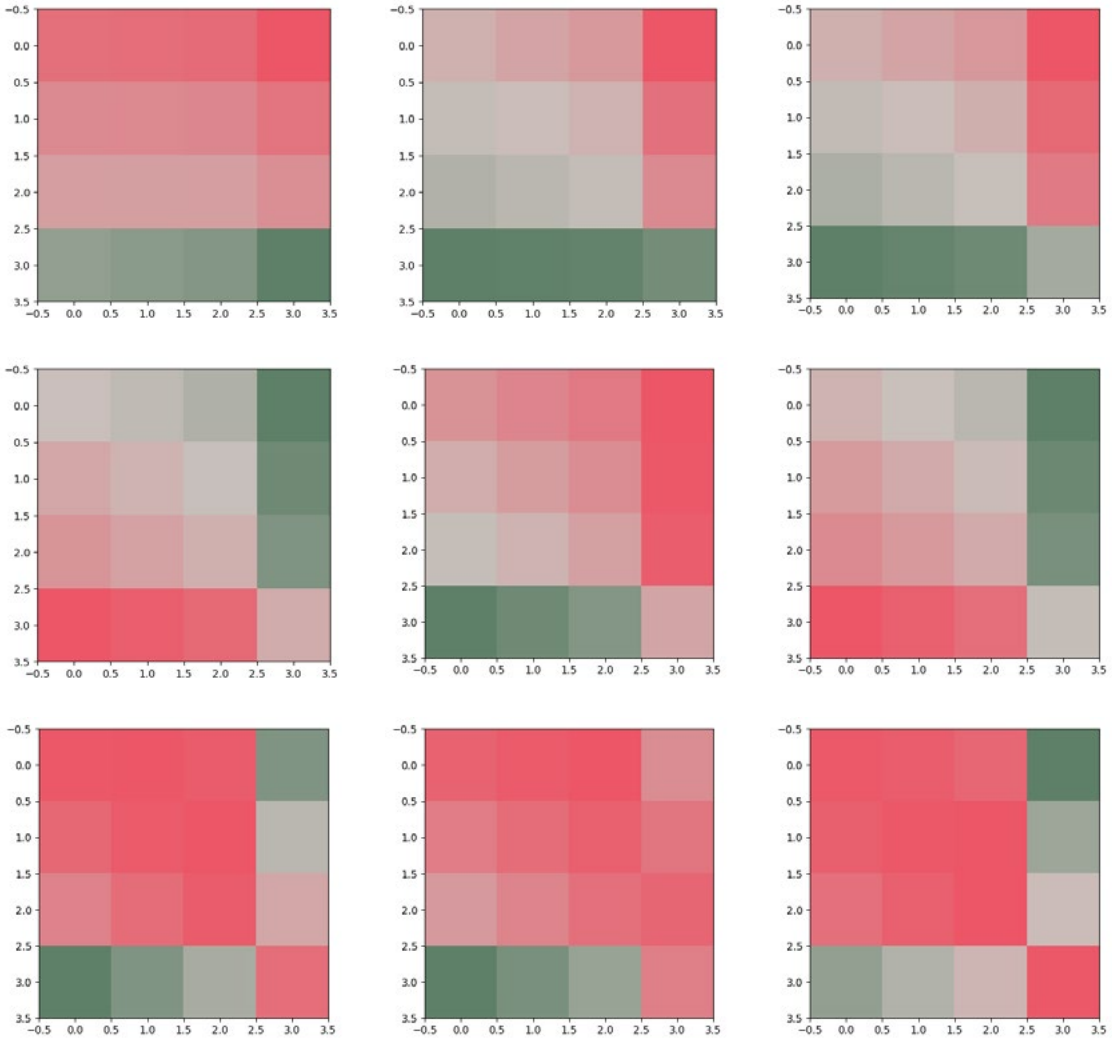
* Last epoch data graph

2. Plot the U-Matrix of the SOM at epoch 4



9. Activate the SOM

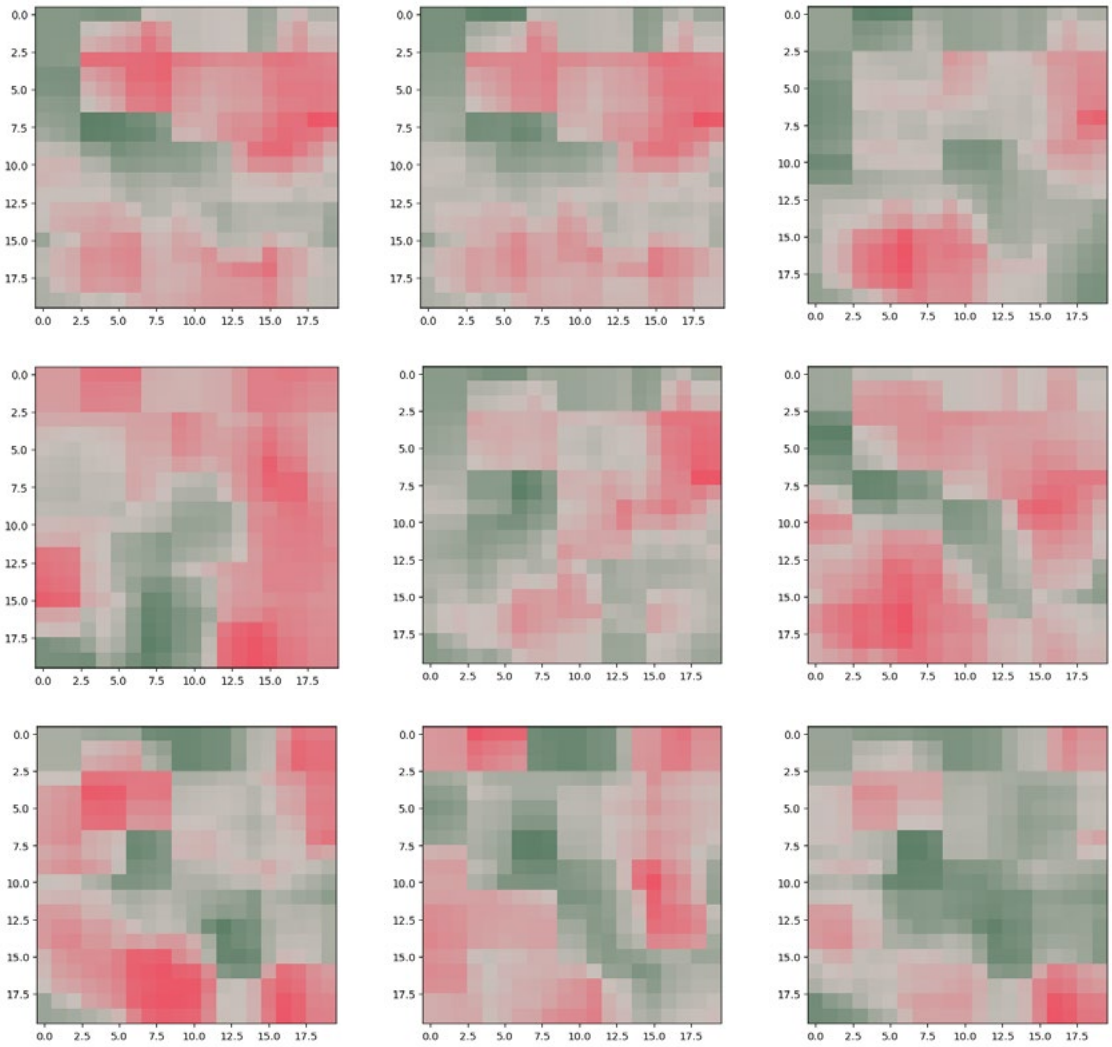




* SOM activations

TEXT 20

1. Proceed with the process explained earlier and activate the SOM

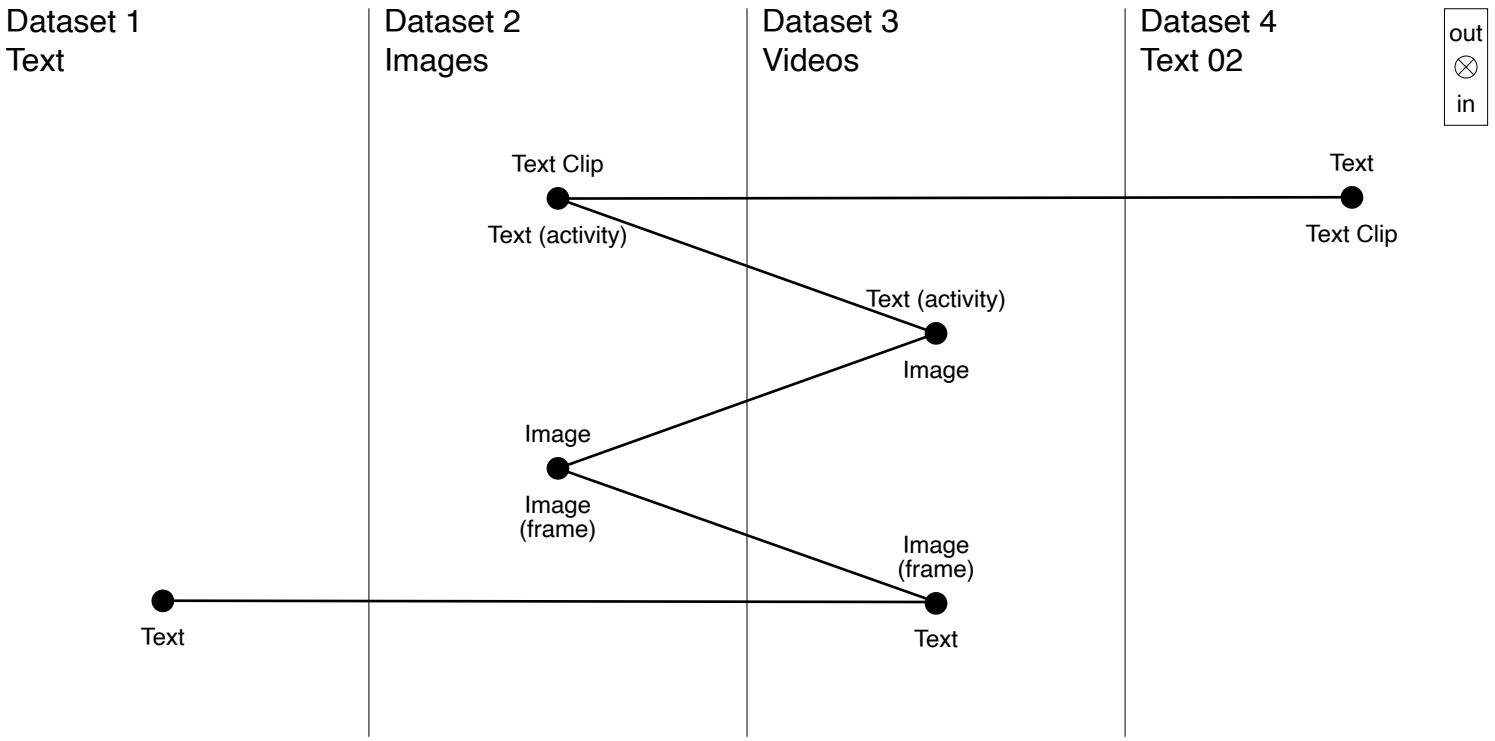


* SOM activations

III. SEARCH ENGINE

COLUMNS ATLAS

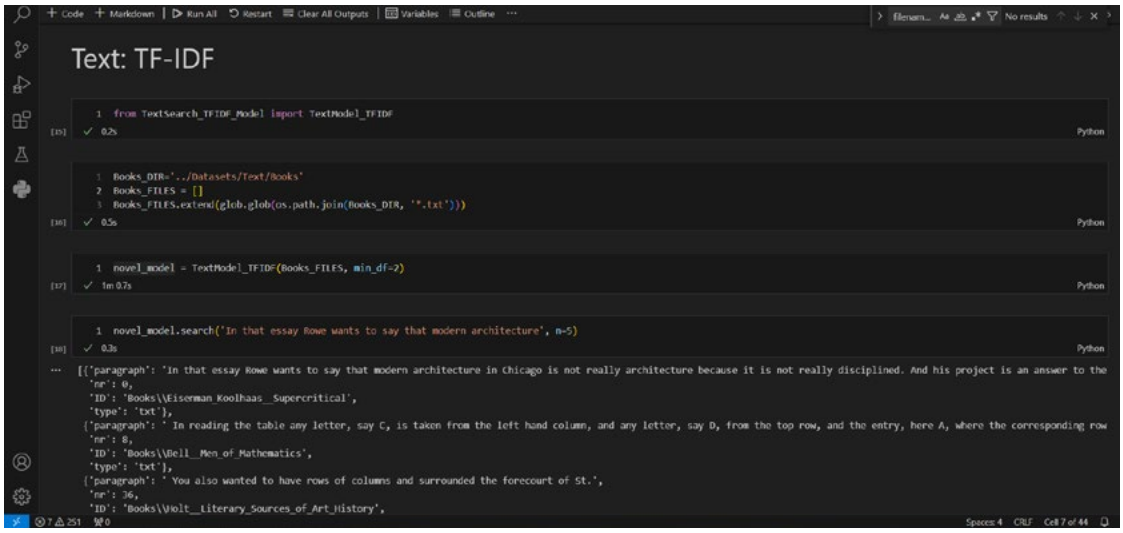
Ebitiust quidundio. Ut apietur? Quiat aspici digendundit eatur ad et moditat inctet omniatem quunda comnien dantibus, officti oremporum, que et esenimp orest, volo tem ullabor aut re preprem harciatem suntur aut evelles et atempor sim et et lam quid quoditate si non perit, tem eici ut in pellescidel ipsumqu identor simagna musandu cipsanda inveliquatur modisin veliam que dem fugia sequos sus con pro vel mod quid eate ate dolupturist earibus ducilisquis ut andandero molo doluptae quasped molorissus qui cus prest volligendel exerissimet as explignim di dolupta veri cum volorateni dio tet maximpos rem consequod qui acere nim re sim quas que net et voluptatur? Officae venihil lessequo etur alia endam sitatur?



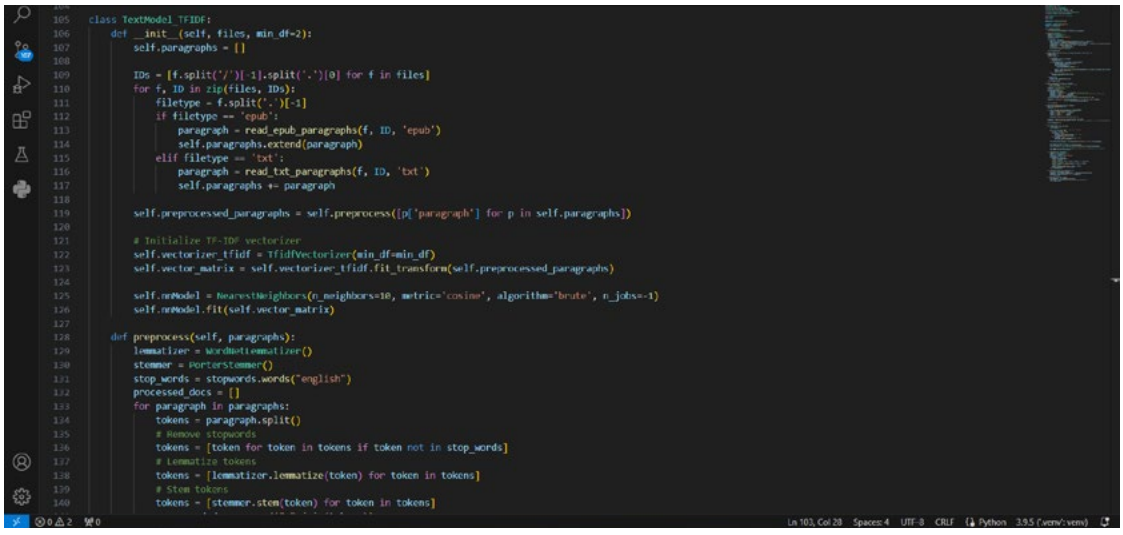
TF-IDF VECTORISATION

Import The Model & Process The Text.

The imported model reads all the files and extracts the paragraphs. Then, the text is preprocessed which includes removing the stopwords and lemmatizing. Finally, the model vectorizes the text.



* TF-IDF

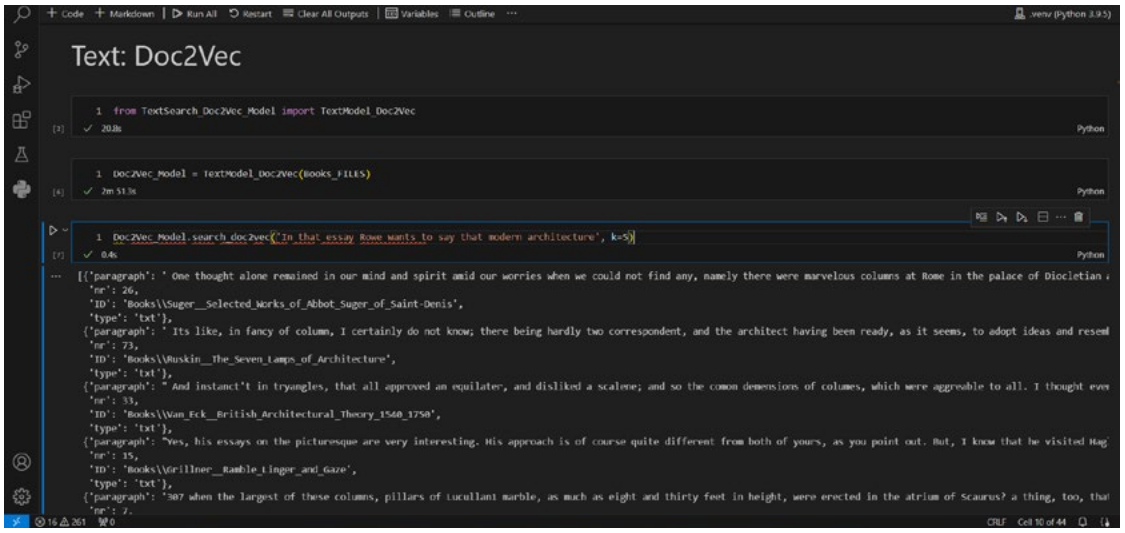


* Snippet of TextSearch_TFIDF_Model.py

DOC2VEC VECTORISATION

Import The Model & Process The Text.

The imported model reads all the files and extracts the paragraphs. Then, the text is preprocessed which includes removing the stopwords and lemmatizing. Finally, the model vectorizes the text.

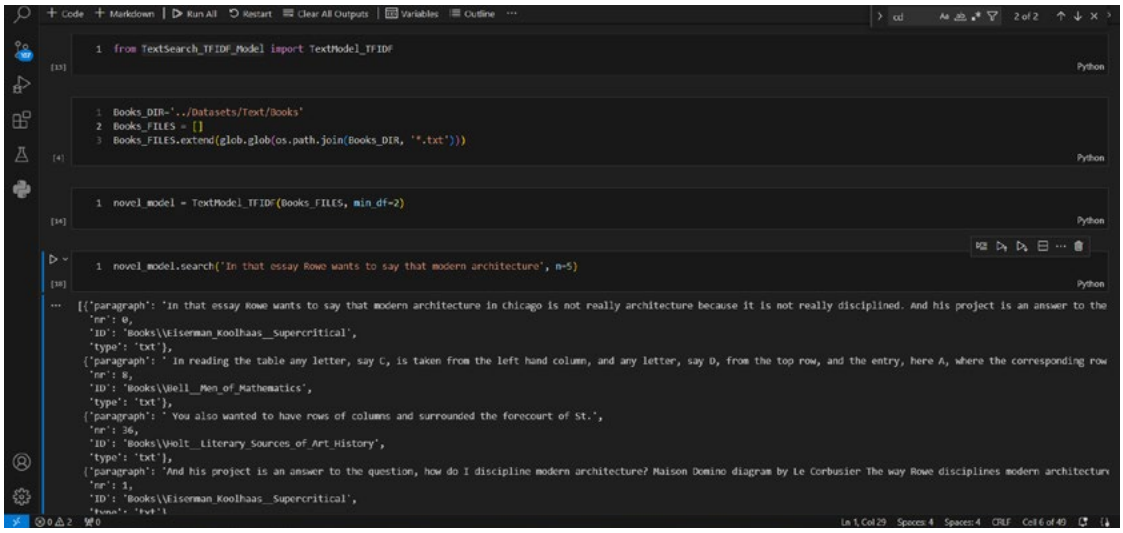


* Doc2Vec

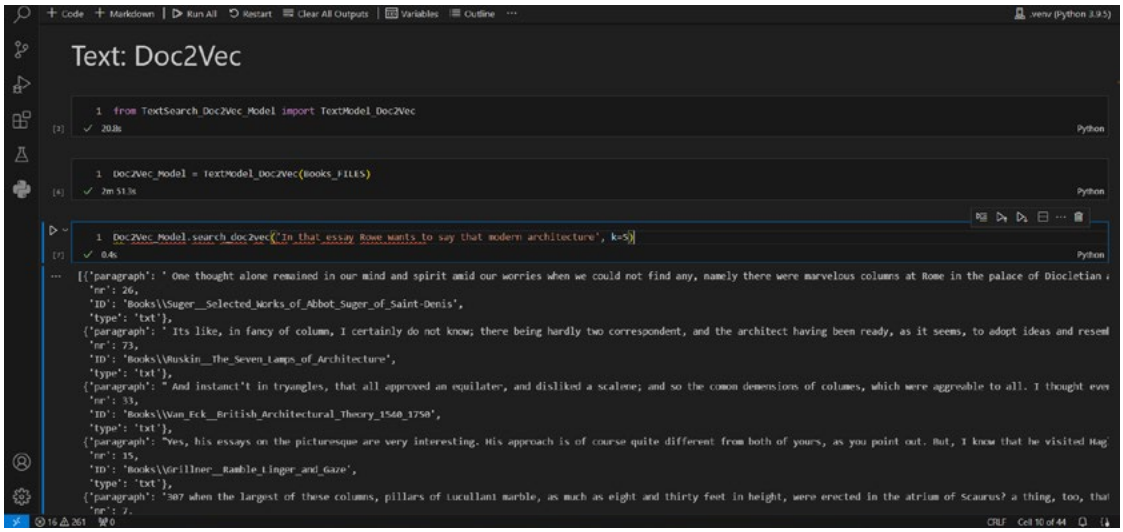
* Snippet of TextSearch_Doc2Vec_Model.py

CONCLUSION

To examine both models, I am using the same query to compare the outputs. The used query is a segment extracted from a sentence from the data processed by both models. The TF-IDF model was able to respond with the full sentence of the query. However, the full sentence was not present in the best five matches of the Doc2Vec model. Also, throughout the tests, Doc2Vec model responses were not consistent. This is because each time the model runs, it selects different segments from the data for training, resulting in inconsistent responses for the same query. Yet, whether this is a disadvantage or not depends on the context.

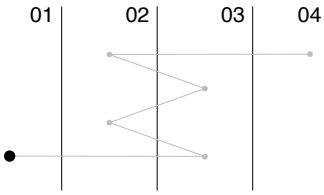


* TF-IDF response



* Doc2Vec response

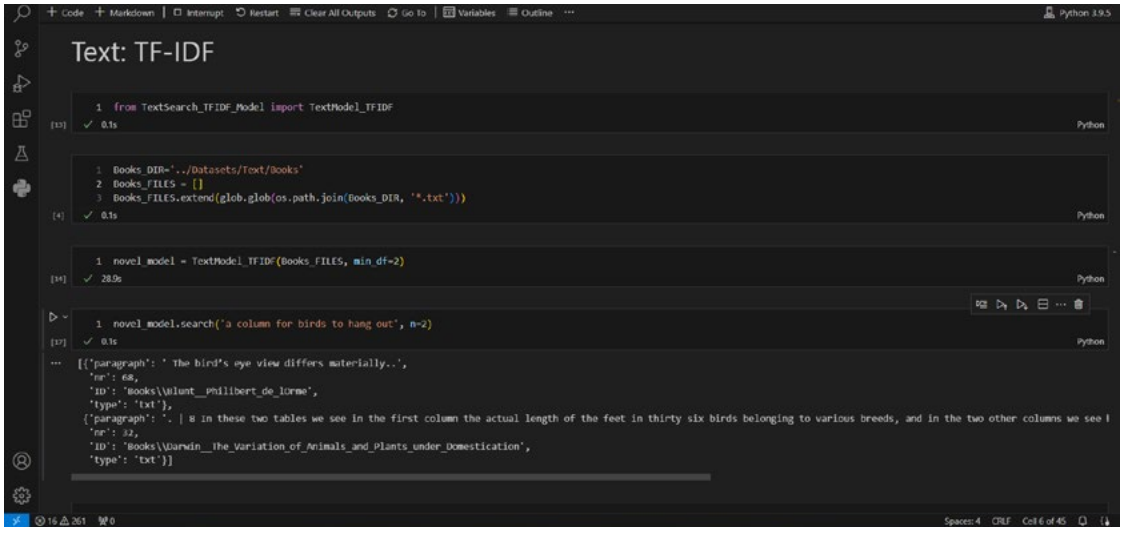
TEXT SEARCH



Use dataset 1 (text) as an input for the TF-IDF vectorizing model. Then seach in dataset 1 using the query “‘a column for birds to hang out’.

Output:

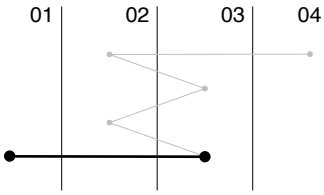
{‘paragraph’: ‘. | 8 In these two tables we see in the first column the actual length of the feet in thirty six birds belonging to various breeds, and in the two other columns we see by how much the feet are too shore or too long, according to the size of bird, in comparison with the rock pigeon.’,
‘nr’: 32,
‘ID’:‘Books\\Darwin__The_Variation_of_Animals_and_Plants_under_Domestication’,
‘type’: ‘txt’}]

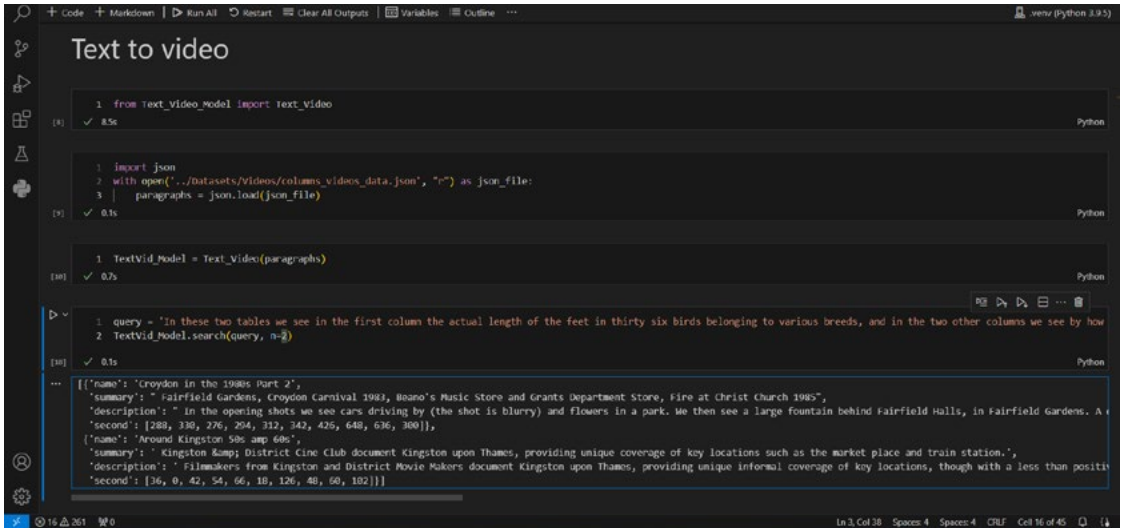


* Text search

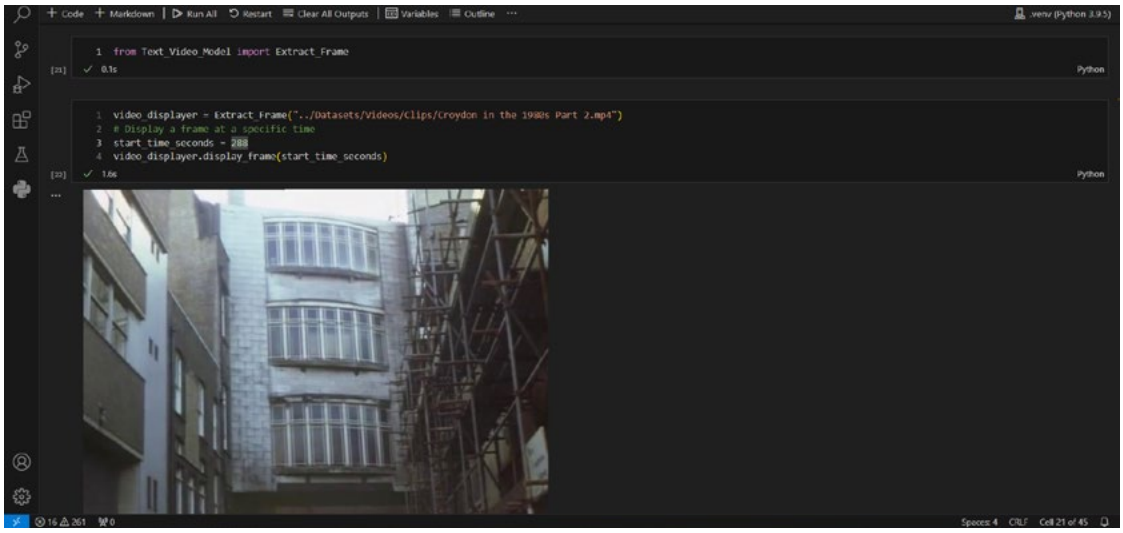
TEXT TO VIDEO

Select the second response of the previous step. Input the response into the Text_Video model which searches in the text data linked to dataset 3 (videos). The response is a list that contains the film name that is closest to the query, a summary of the video, a description, and the seconds that were extracted by the columns detection model. Then use Display_Clip to display the clip using the output of the previous step. Eventually, extract the frame for the exact second.

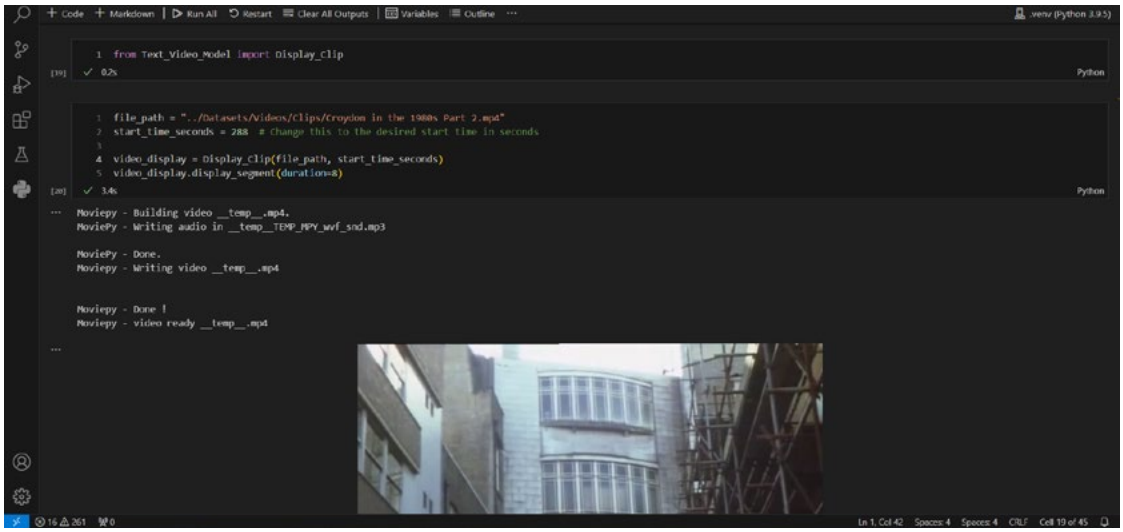




* Text search



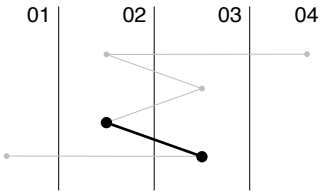
* Display the clip by inputing the clip name and specify the second

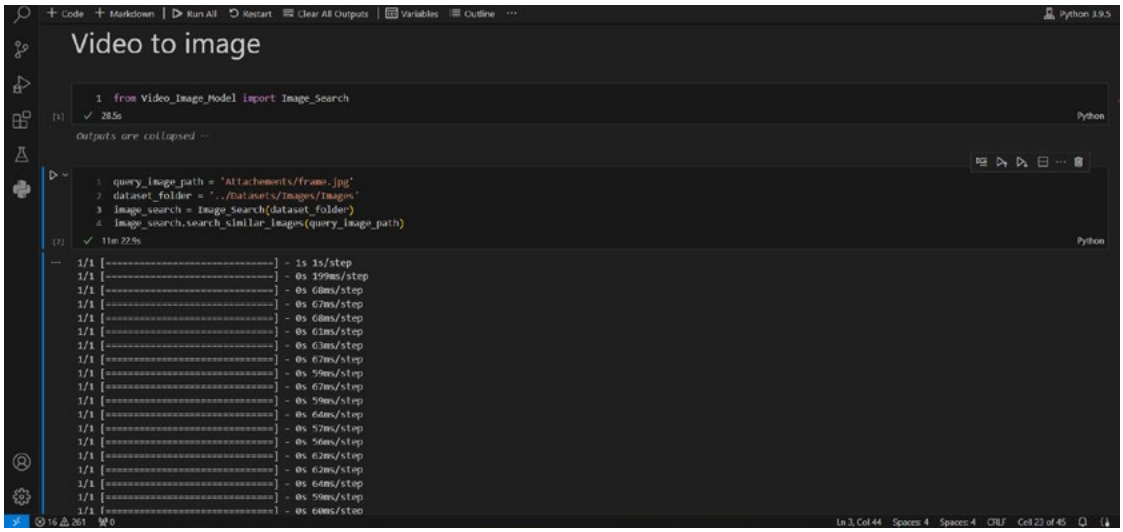


* Extract the frame

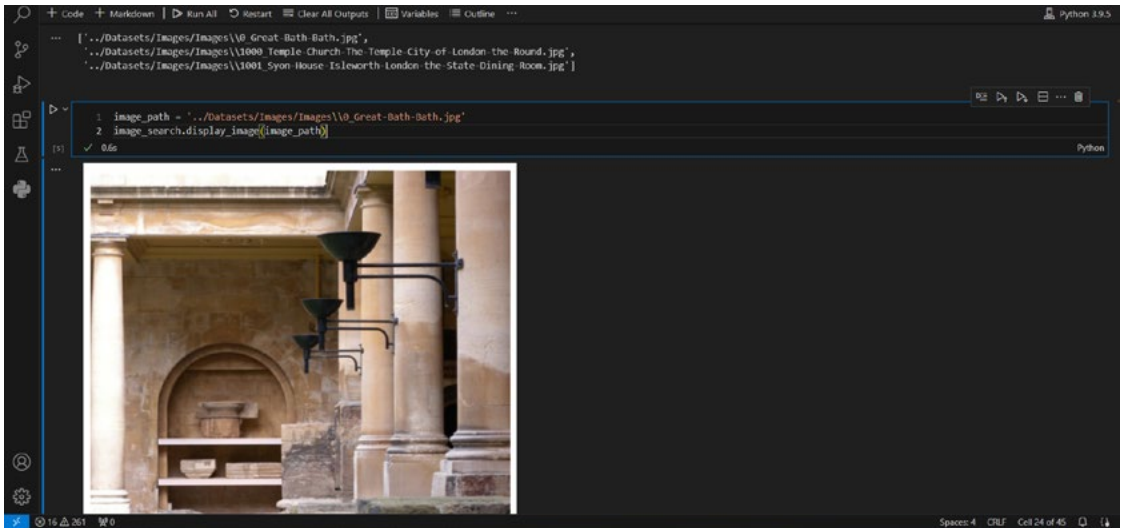
VIDEO TO IMAGE

Use the frame extracted from the previous step to search in dataset 02 (images) and get the best matching image. The model mainly extracts features of the query image and dataset 02 using mobilenet model. Then it compares the features to find the best matches. Eventually, it prints the path of the three best matching images which is displayed in the next block.





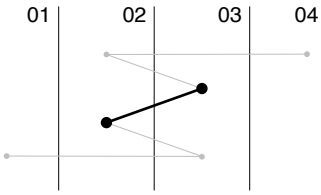
* Images search

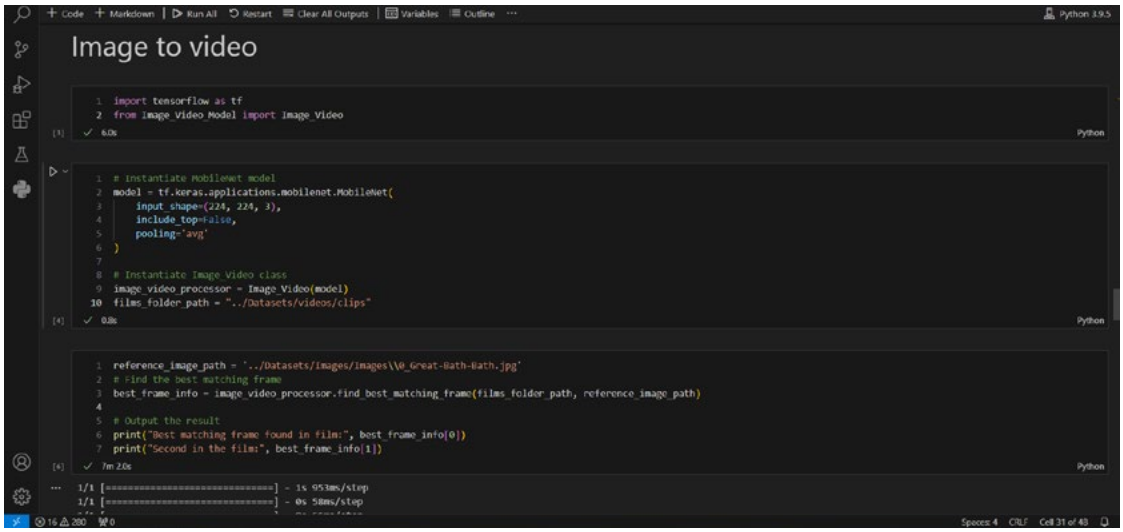


* The path of the best matching image and displaying it

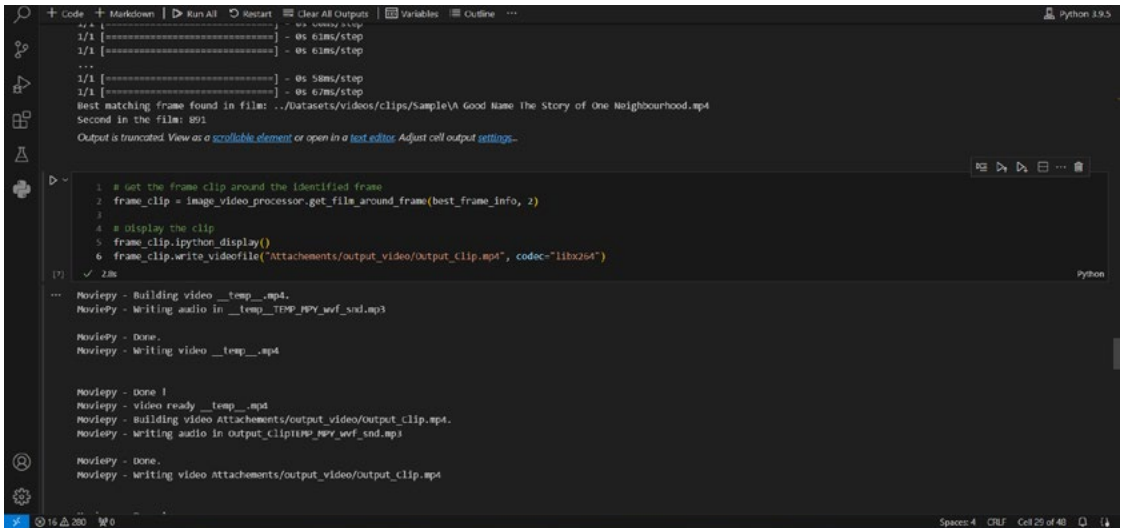
IMAGE TO VIDEO

At this step, the output image of the previous step is used as a query to search in dataset 03 (videos). The output of this step is a film name and the best-matching second in the film. Then, these two outputs are used to save the clip.





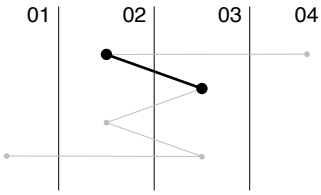
* Using the extracted frame to search in dataset 3

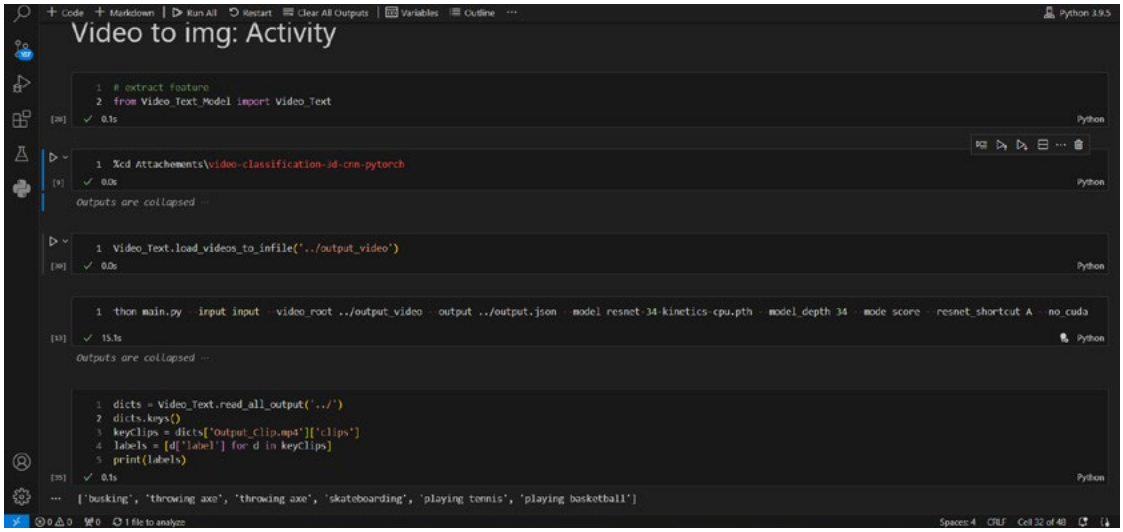


* The output of the film search is the best matching film path and best matching second of the film. Then save the clip around the best matching second.

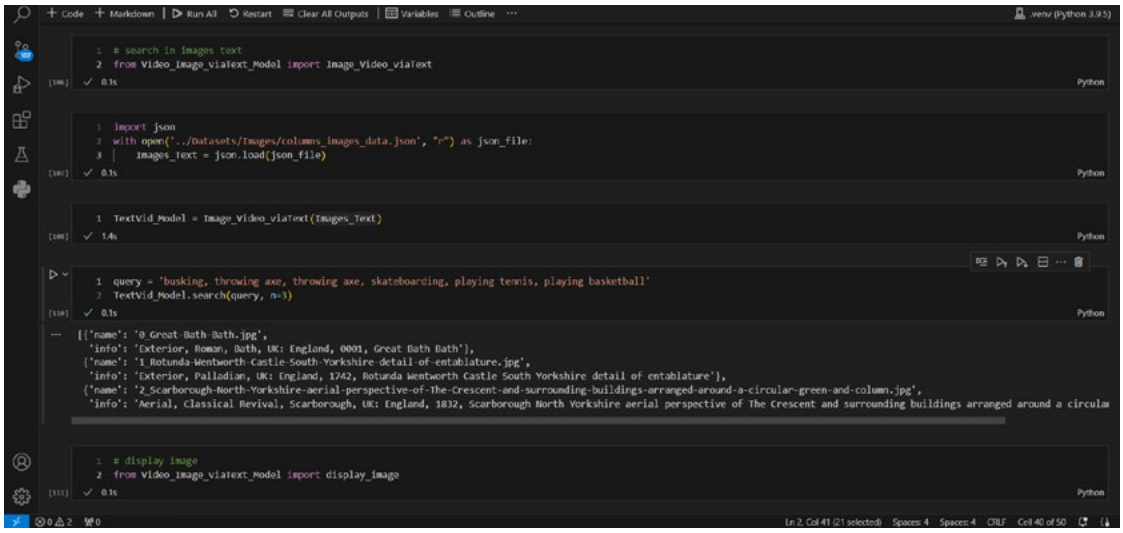
VIDEO TO IMAGE: ACTIVITY

The features (activity) of the output clip from the previous step are extracted and then used as a query to search in the text data linked to dataset 02 (images). The output of this step is an image.

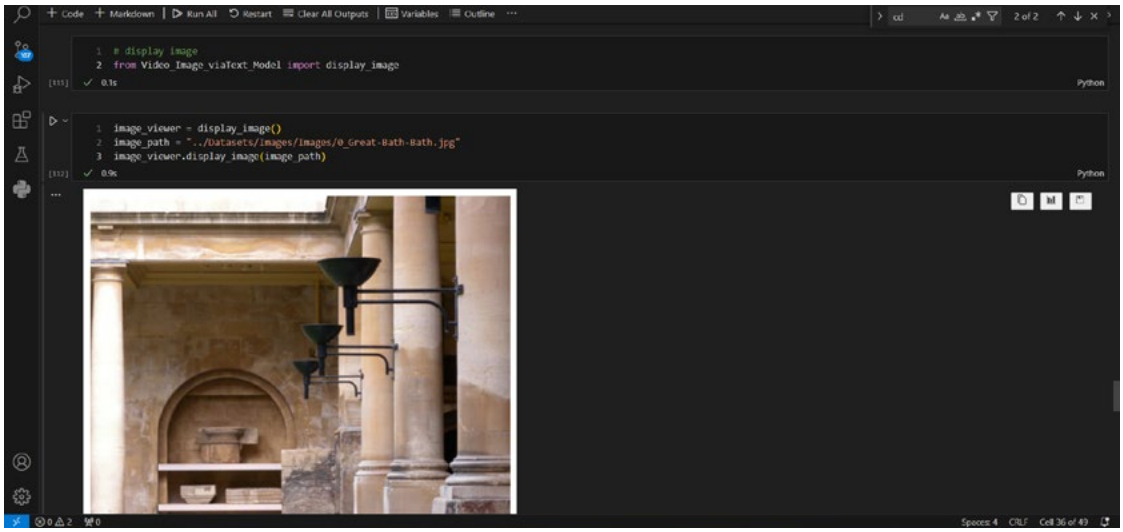




* Extract the activity



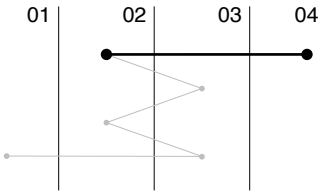
* Use the activties to search in the text data linked to dataset 02 (images)

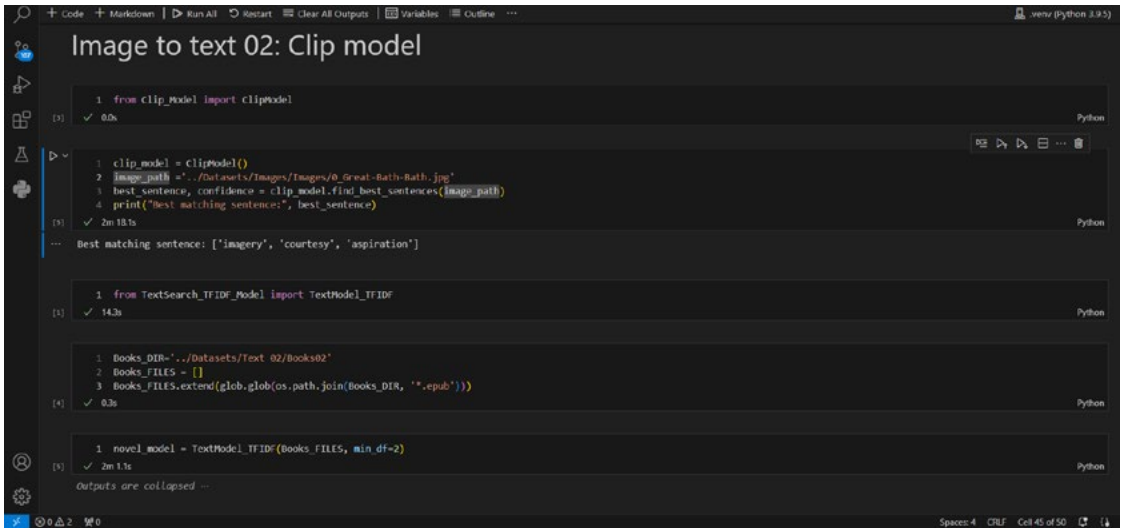


* Best matching image

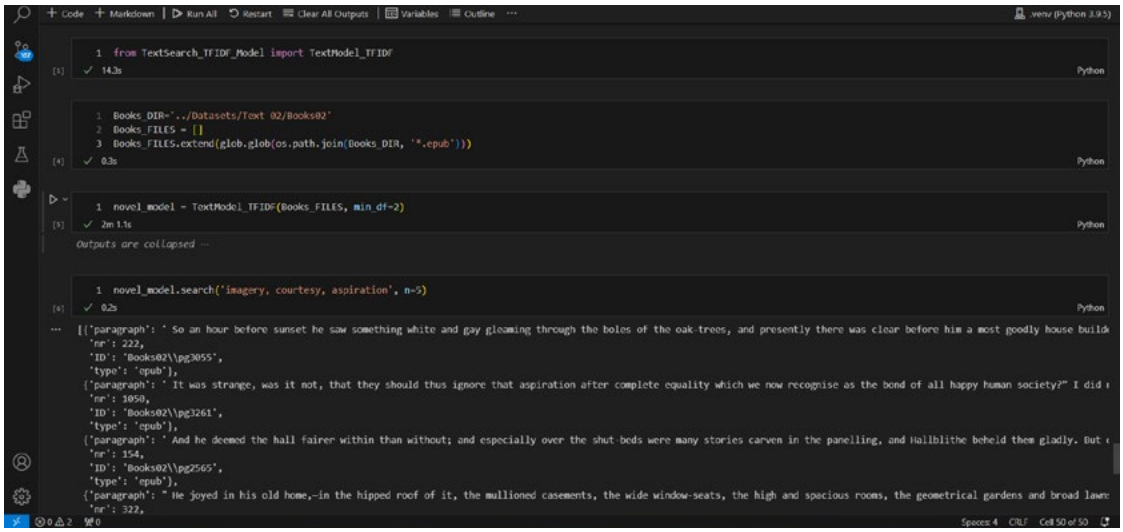
IMAGE TO TEXT 02: CLIP MODEL

Use Clip model to describe the image using the 3 best-matching words from the dictionary. Then use TextModel_TFIDF to vectorize and process dataset 04 (text). Use the output of clip model as a query in dataset 04.





* Labeling the image using Clip model



* Search in dataset 04

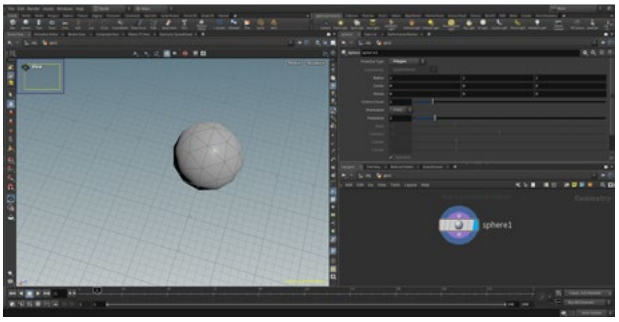
HOUDINI

IV. HOUDINI FUNDAMENTALS

PROCEDURAL ANIMATION

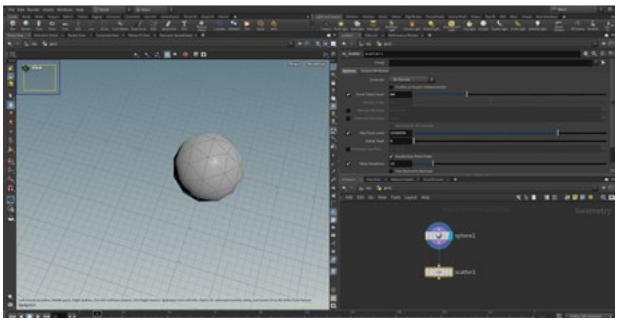
01 PYTHON NODE

Create a geomtry, then create a sphere and set to polygon with a frequency of 2



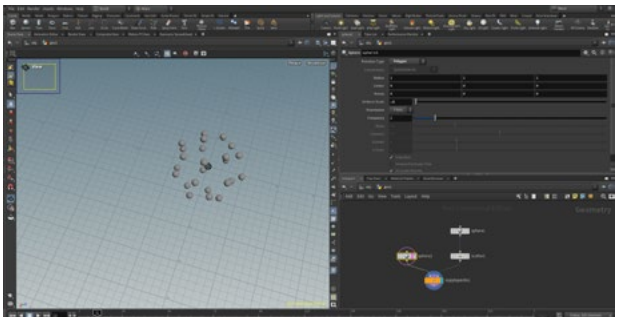
02 SCATTER

Add scatter to scatter some points on the sphere and set to 24



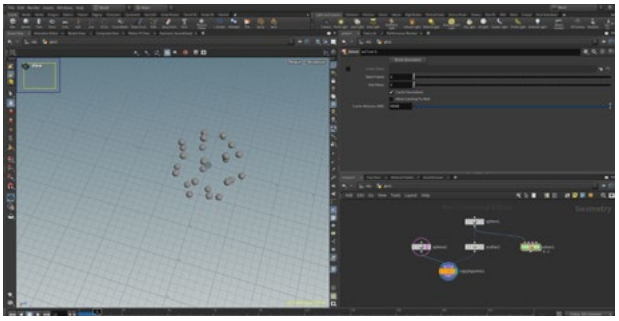
03 COPY TO POINTS, ADD A SPHERE

set the sphere to polygons with a frequency of 2, and scale it to 0.1. Then, connect the scatter and the sphere with the copy to points



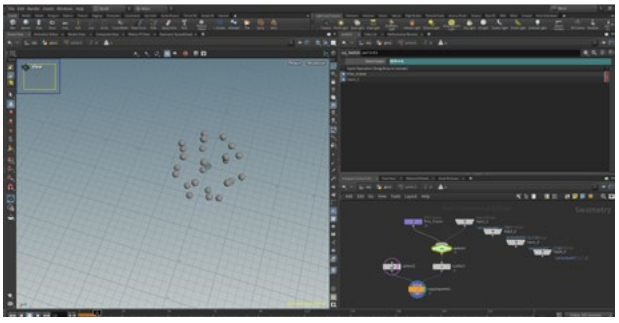
04 ADD A SOLVER

To create the animation and operate on the previous frame geometry



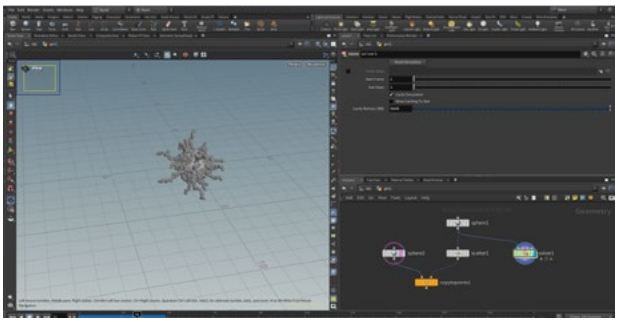
05 ADD A SWITCH

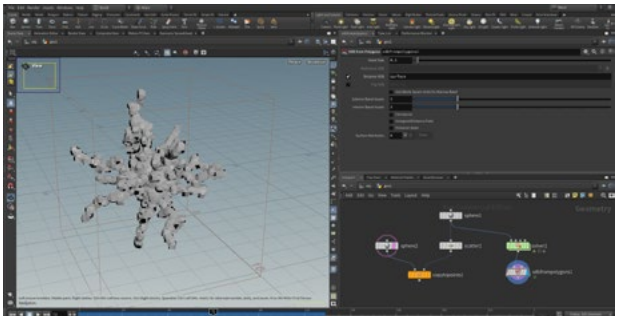
Inside the solver, add a switch and add a condition if the frame equals 1 switch to input_1, then paste the 3 nodes that were created in the previous steps and connect the switch to the scatter.



07 ADD A VDB FROM POLYGONS

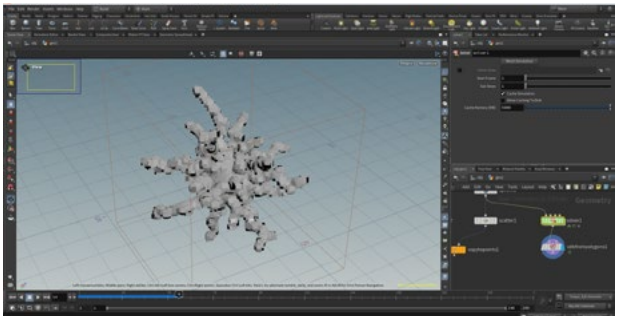
Add a vdb from polygons to add a pixelated effect on the spheres and connect it to the solver

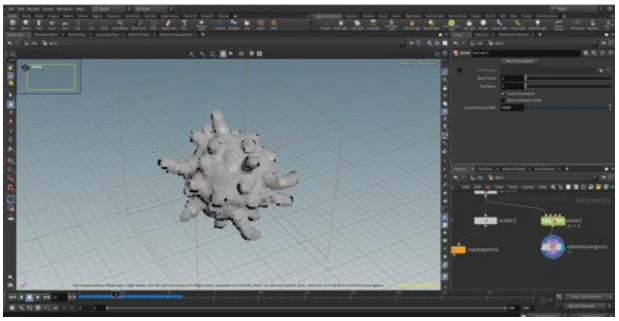


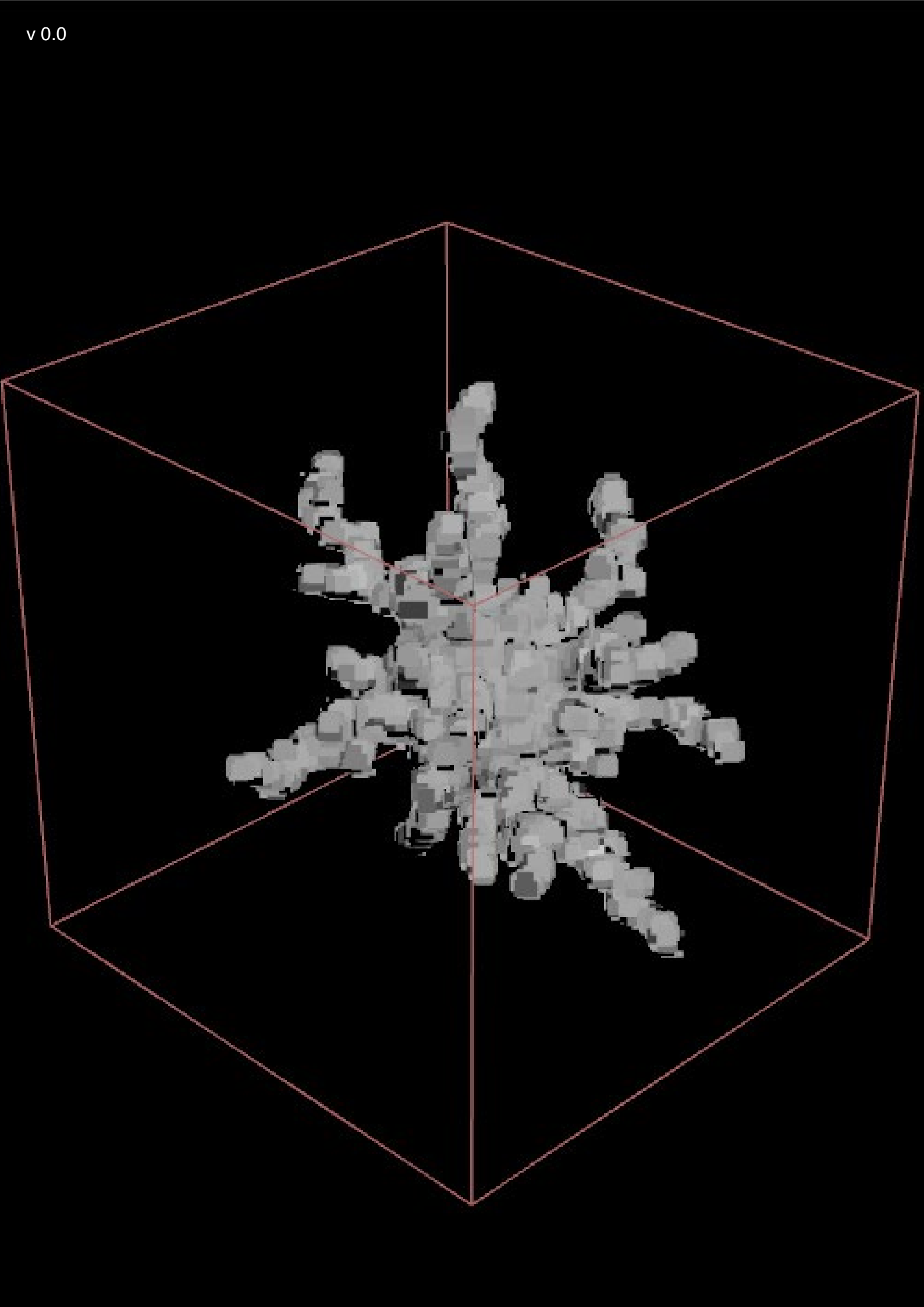


08 FINAL ANIMATION

2 different frames from the animation



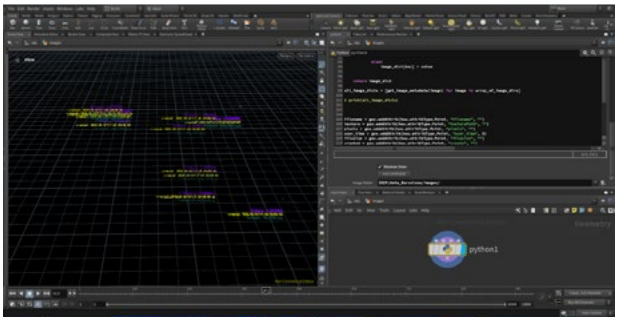



v 0.0

V. VISUALIZING GPS & IMAGE METADATA

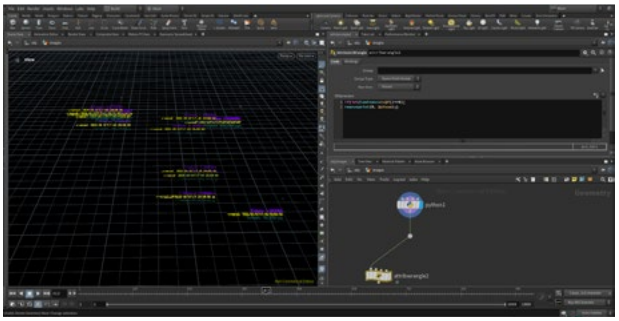
01 PYTHON NODE

Create a Python SOP and link it to the images folder



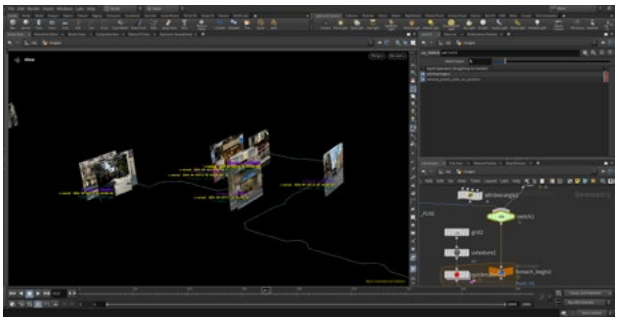
02 ATTRIBUTE WRANGLE

Remove unwanted images



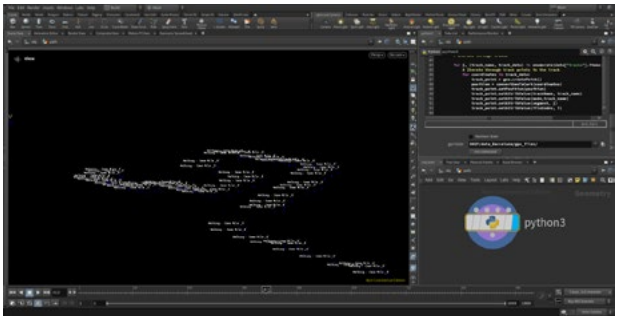
03 SWITCH SOP

Oriente images toward the camera



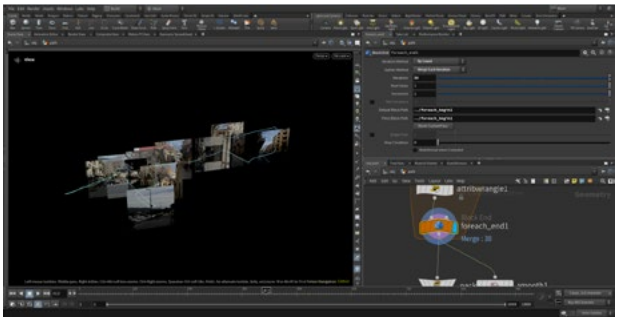
04 PYTHON NODE

Create a Python SOP and link it to the path gpx file



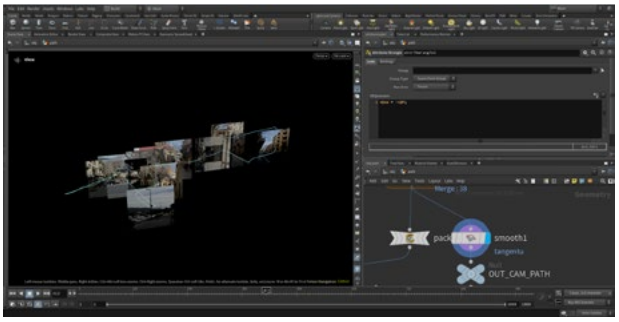
05 ADD A FOREACH

Create a line out of the points



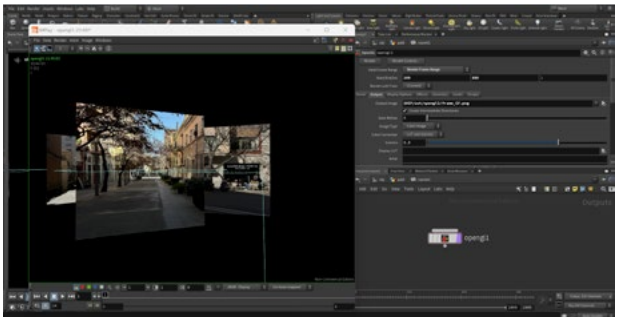
06 SMOOTH SOP

Create a geomtry, then create a sphere and set to polygon with a frequency of 2



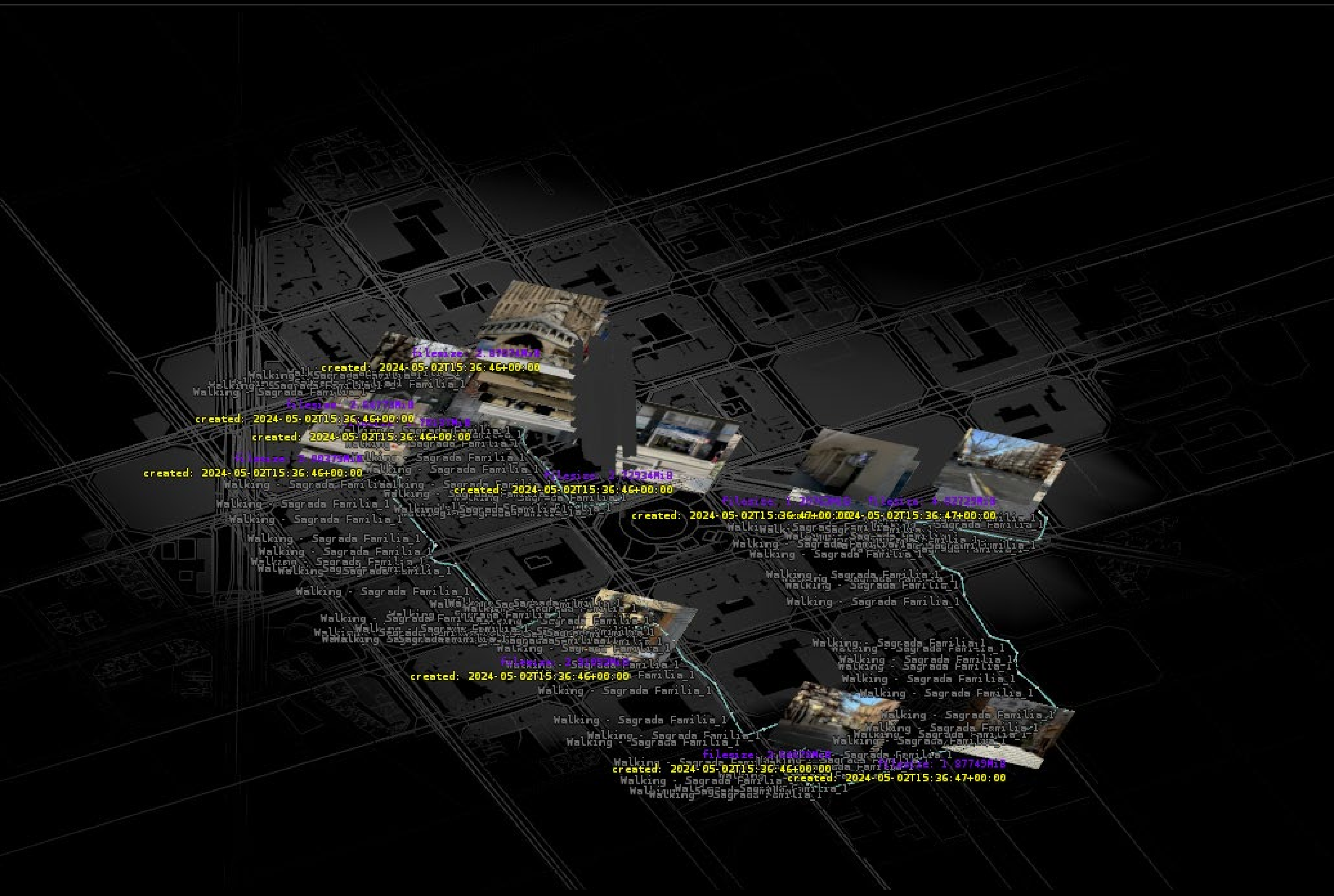
07 SCATTER

smooth the created line to set it as a reference for the camera path



created: 2024-05-02T15:36:45+00:00
created: 2024-05-02T15:36:45+00:00
created: 2024-05-02T15:36:45+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:47+00:00

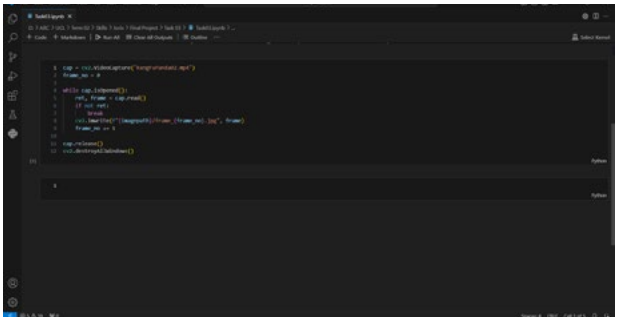

Ortho
Walking - Sarria_3
created: 2024-05-02T15:36:45+00:00


Walking - Sagrada Familia_1
created: 2024-05-02T15:36:46+00:00
created: 2024-05-02T15:36:47+00:00

VI. VIDEO TO 3D MODEL

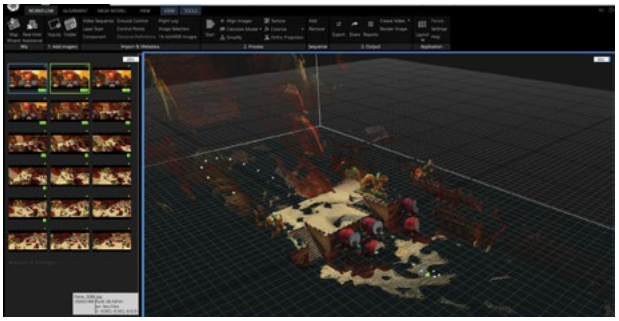
01 EXTRACT FRAMES

Use the Python code to extract the frames from the mp4 videos.



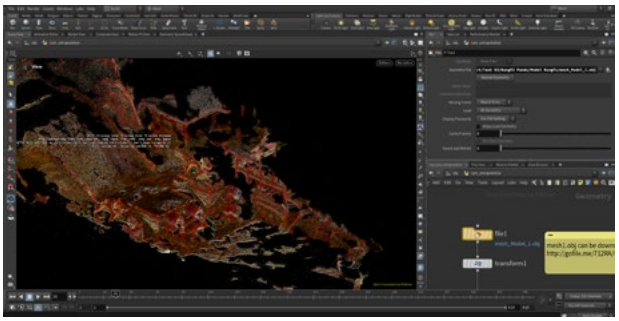
02 CONSTRUCT THE 3D MODE

Import the frames CaptureReality and reconstruct the scenes. Then export as obj.



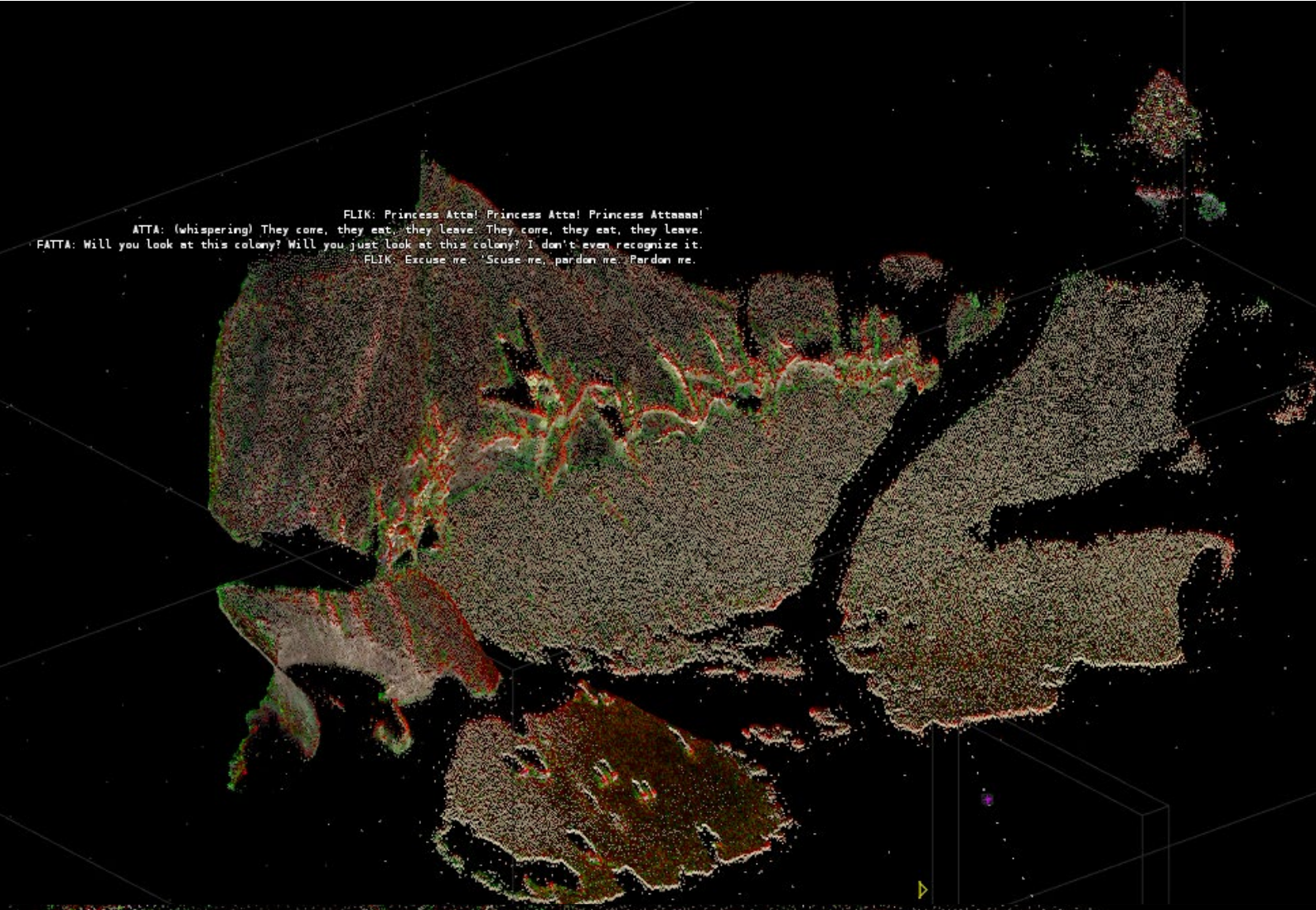
03 IMPORT TO HOUDINI

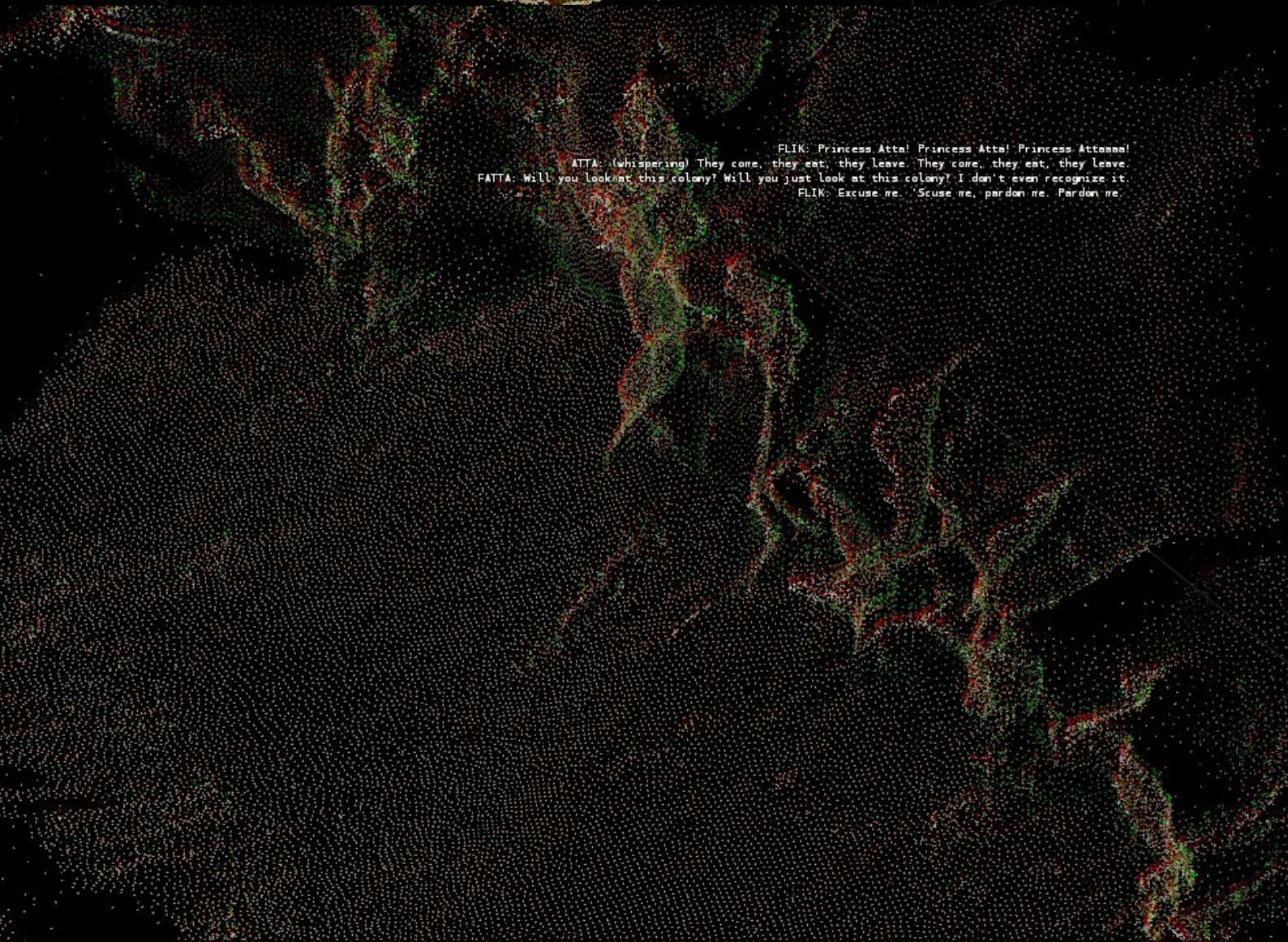
Import the model to Houdini using a file SOP. Then add captions and annotations. Match the camera path and export a video.




FLIK: Princess Atta! Princess Atta! Princess Attaaaa!
ATTA: (whispering) They come, they eat, they leave. They come, they eat, they leave.
FATTA: Will you look at this colony? Will you just look at this colony? I don't even recognize it.
FLIK: Excuse me. 'Scuse me, pardon me. Pardon me.


FLIK: Princess Atta! Princess Atta! Princess Attaaaa!
They come, they eat, they leave. They come, they eat, they leave.
ony? Will you just look at this colony? I don't even recognize it.
FLIK: Excuse me. 'Scuse me, pardon me. Pardon me.

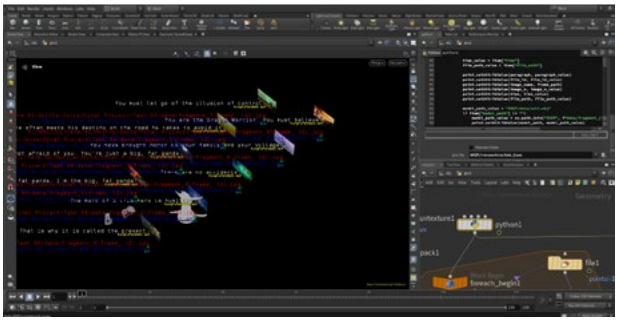

FLIK: Princess Atta! Princess Atta! Princess Attaaaa!
ATTA: (whispering) They come, they eat, they leave. They come, they eat, they leave.
FATTA: Will you look at this colony? Will you just look at this colony? I don't even recognize it.
FLIK: Excuse me. 'Scuse me, pardon me. Pardon me.


FLIK: Princess Atta! Princess Atta! Princess Attaaaa!
ATTA: (whispering) They come, they eat, they leave. They come, they eat, they leave.
FATTA: Will you lookmat this colony? Will you just look at this colony? I don't even recognize it.
FLIK: Excuse me. 'Scuse me, pardon me. Pardon me.

VII. VISUALIZING JSON IN HOUDINI

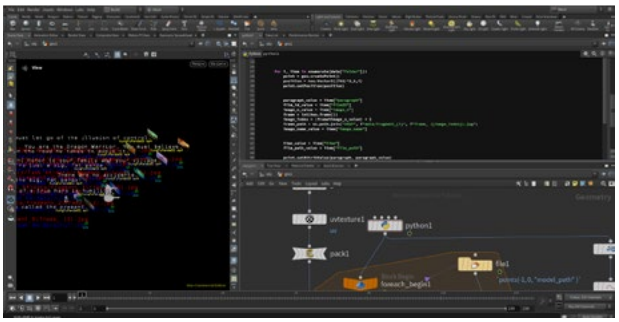
01 PYTHON SOP

Link the json file to the python SOP



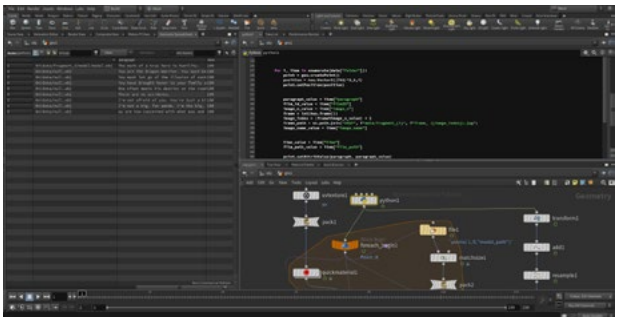
02 IMPORT FRAMES

Through the python SOP code, load the frames.



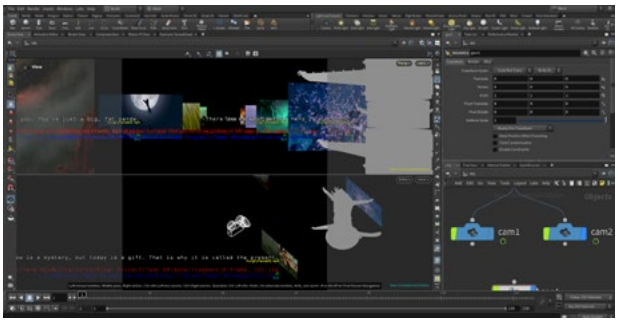
03 IMPORT A 3D MODEL

Based on the json data import the model.



03 EXPORT THE VIDEO

Set two cameras and render the video.

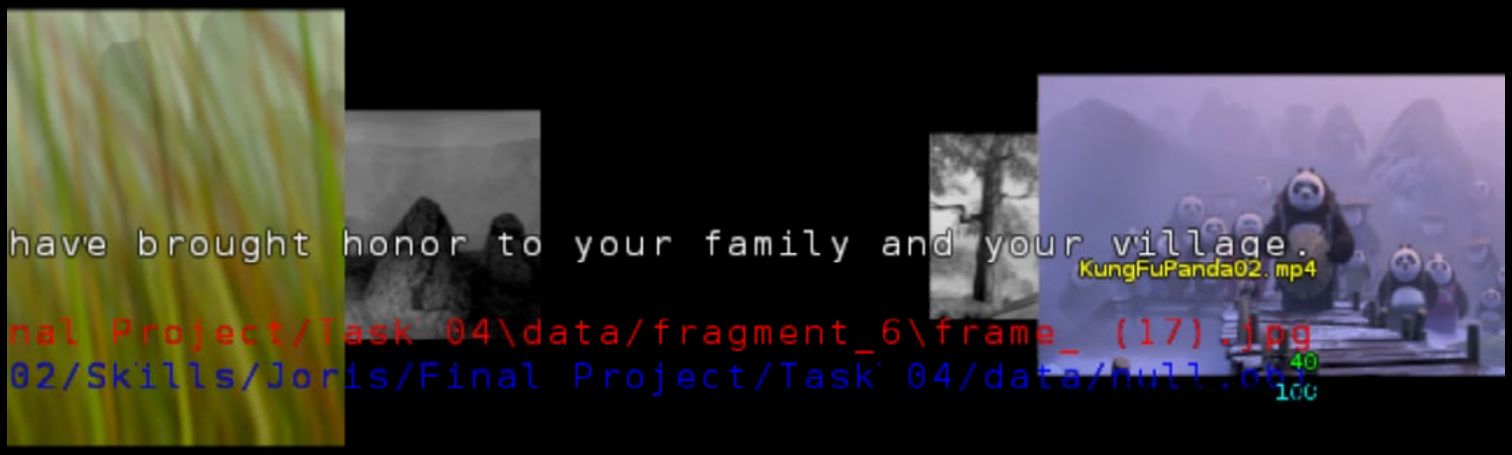


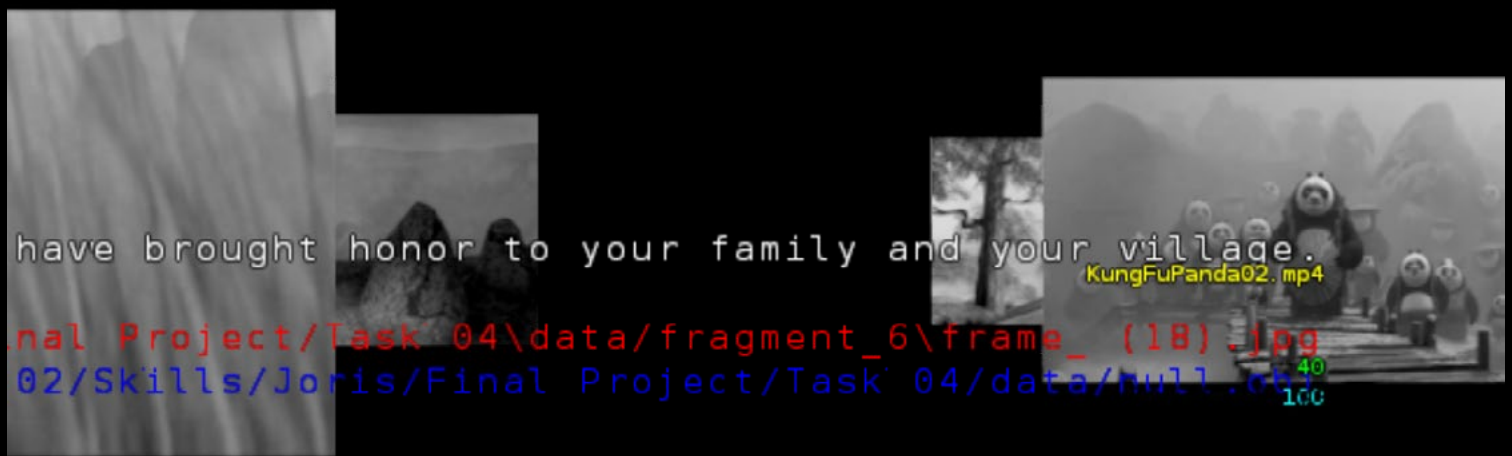

big, fat panda. I'm the big, fat panda!
KungFuPanda02.mp4
Task 04\data/fragment_1\frame_ (14).jpg
ris/Final Project/Task 04/data/null.ob
15
100


, fat panda. I'm the big, fat panda!
KungFuPanda02.mp4
sk 04\data/fragment_1\frame_ (2).jpg
/Final Project/Task 04/data/null.ob
15
100


t panda. I'm the big, fat panda!
KungFuPanda02.mp4
4\data/fragment_1\frame_ (5).jpg
al Project/Task 04/data/null.ob
15
100


have brought honor to your family and your village.
nal Project/Task 04\data/fragment_6\frame_ (16).jpg
02/Skills/Joris/Final Project/Task 04/data/null.tif
KungFuPanda02.mp4
40
100


have brought honor to your family and your village.
nal Project/Task 04\data/fragment_6\frame_ (17).jpg
02/Skills/Joris/Final Project/Task 04/data/null.tif
KungFuPanda02.mp4
40
100


have brought honor to your family and your village.
nal Project/Task 04\data/fragment_6\frame_ (18).jpg
02/Skills/Joris/Final Project/Task 04/data/null.tif
KungFuPanda02.mp4
40
100

COMPLEXITY

VIII. Exercise One

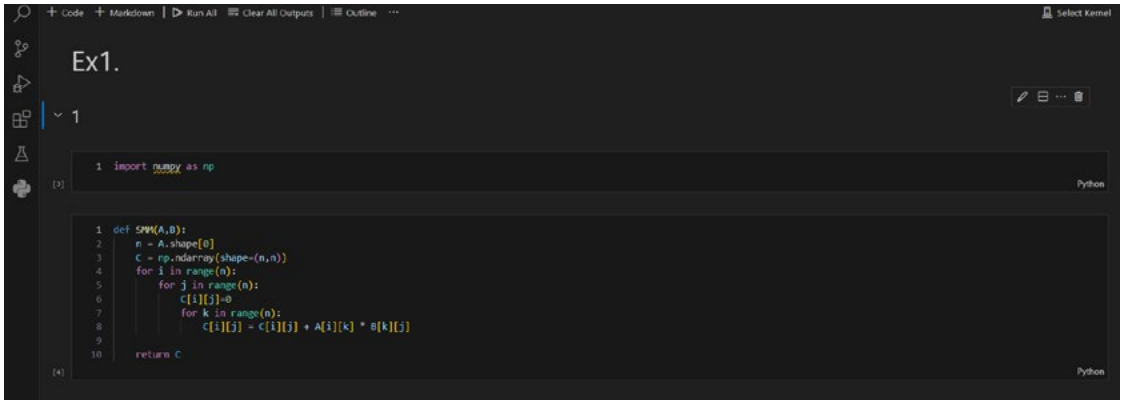
IX. Exercise Two

X. Exercise Three

VIII. EXERCISE ONE

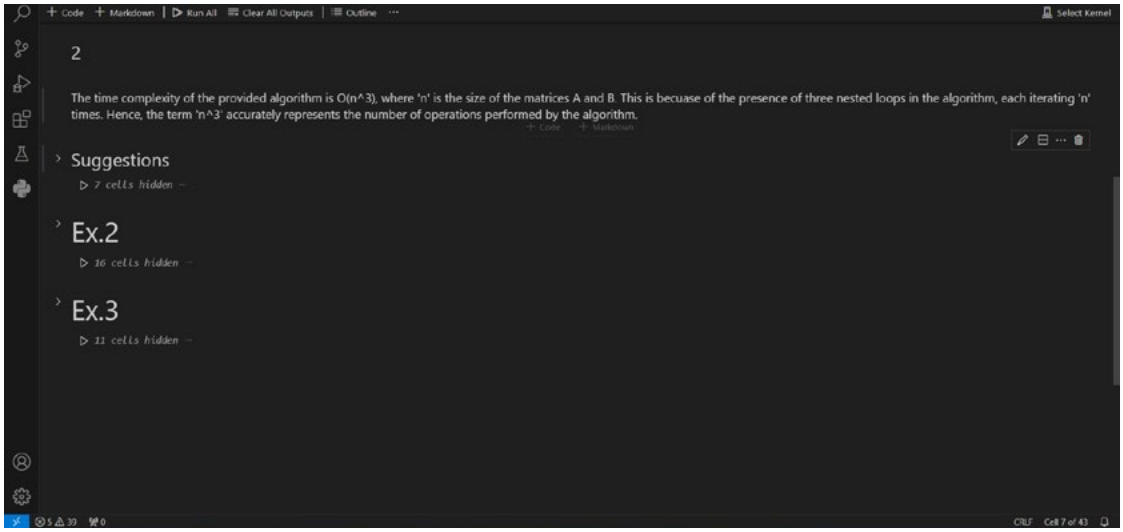
Question 1

Translate the pseudo code to the notebook.



Question 2

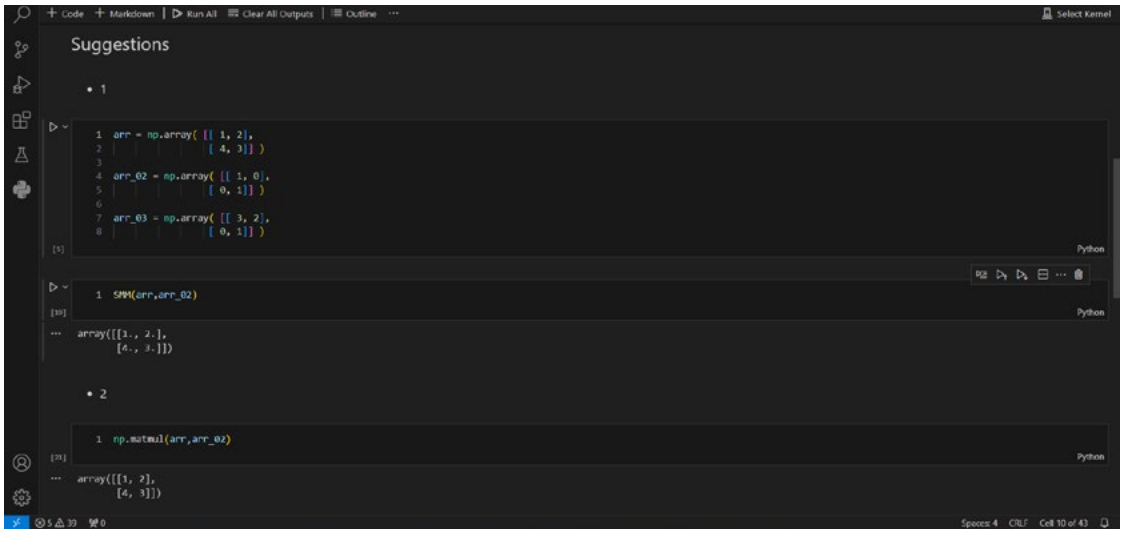
The time complexity of the provided algorithm is O(n^3), where ‘n’ is the size of the matrices A and B. This is becuase of the presence of three nested loops in the algorithm, each iterating ‘n’ times. Hence, the term ‘n^3’ accurately represents the number of operations performed by the algorithm.



Suggestions 1,2

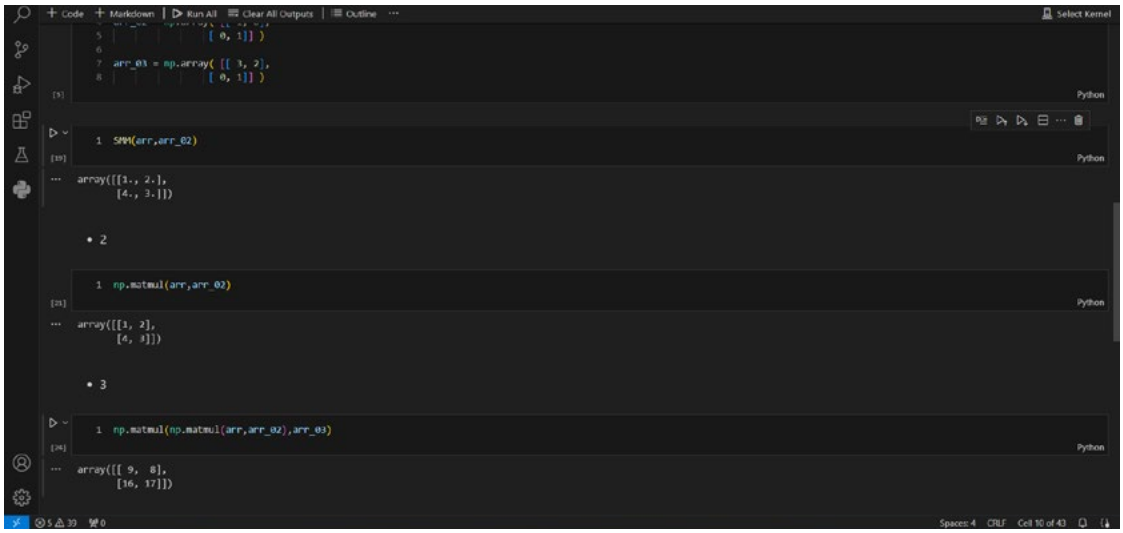
In the first block, I am defining three 2x2 matrices to use for testing. In the second block, I am inputting the defined matrices in the function.

Then, use “np.matmul” to use the predefined function from numpy model.



Suggestions 3

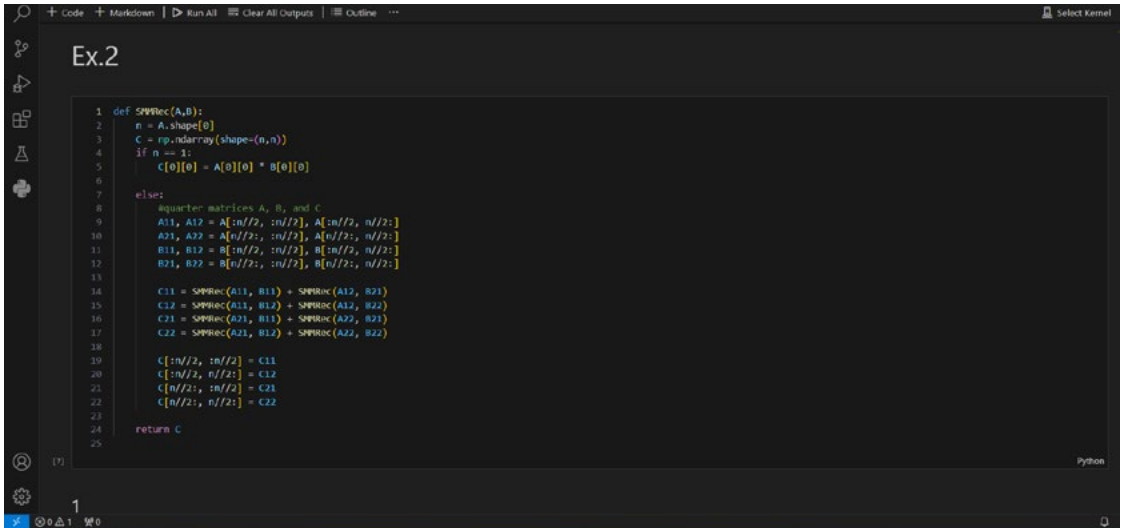
I am using the three defined matrices as in input for the functions.

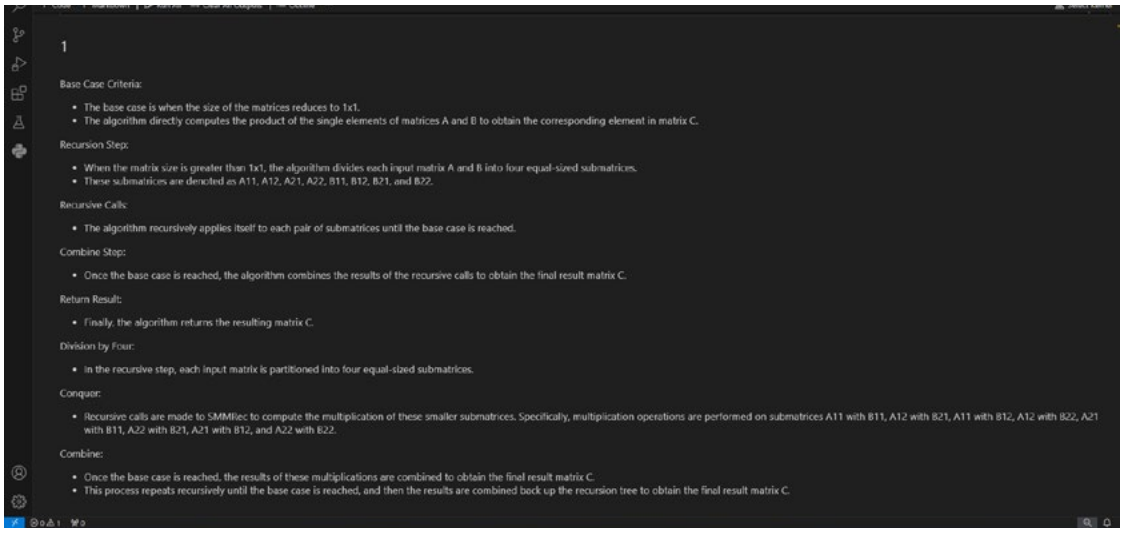


IX. EXERCISE TWO

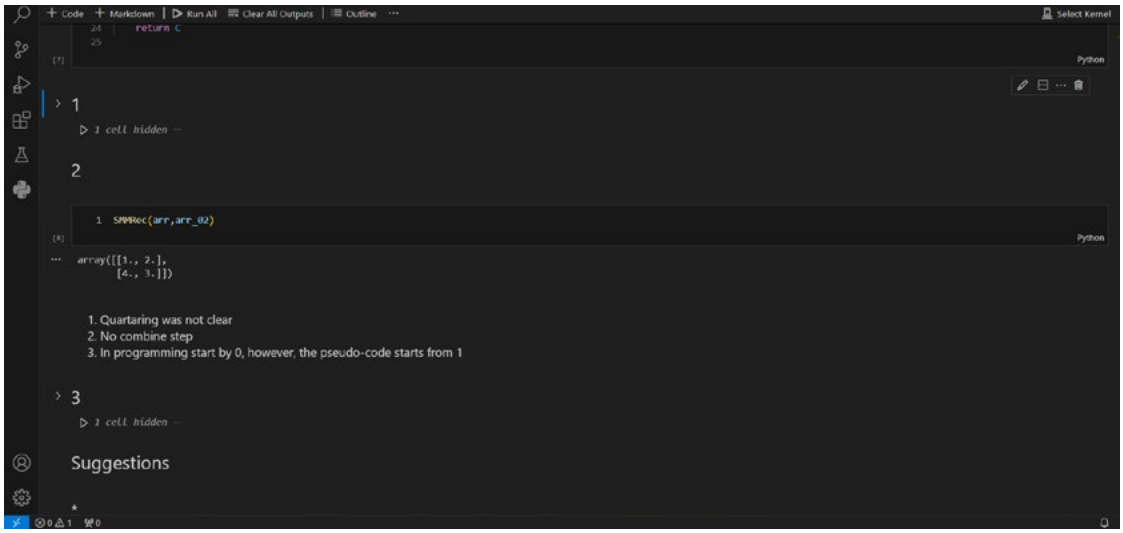
Question 1

Translate the pseudo code to the notebook.

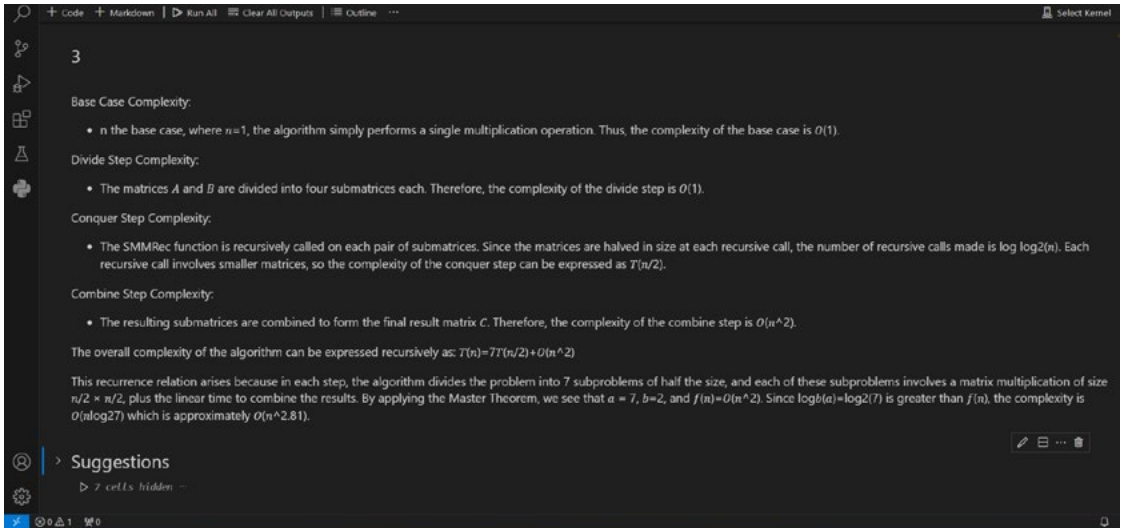




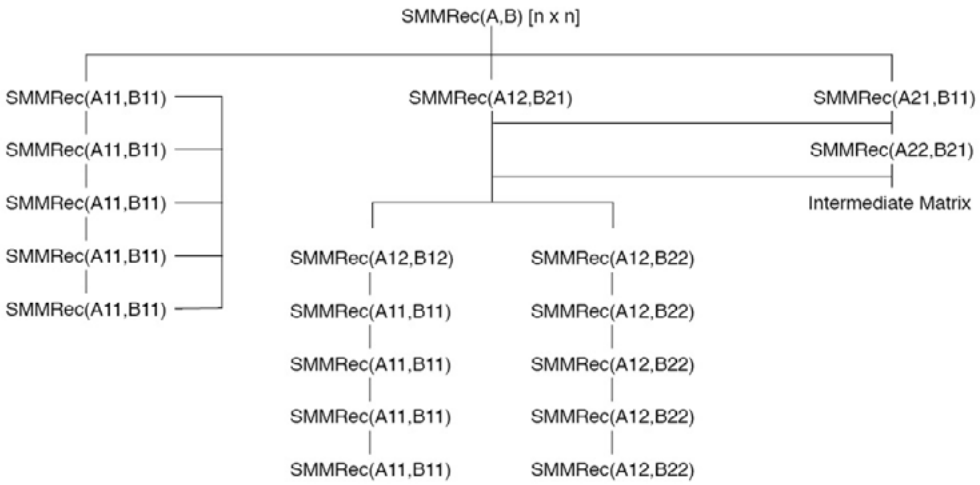
Question 2



Question 3

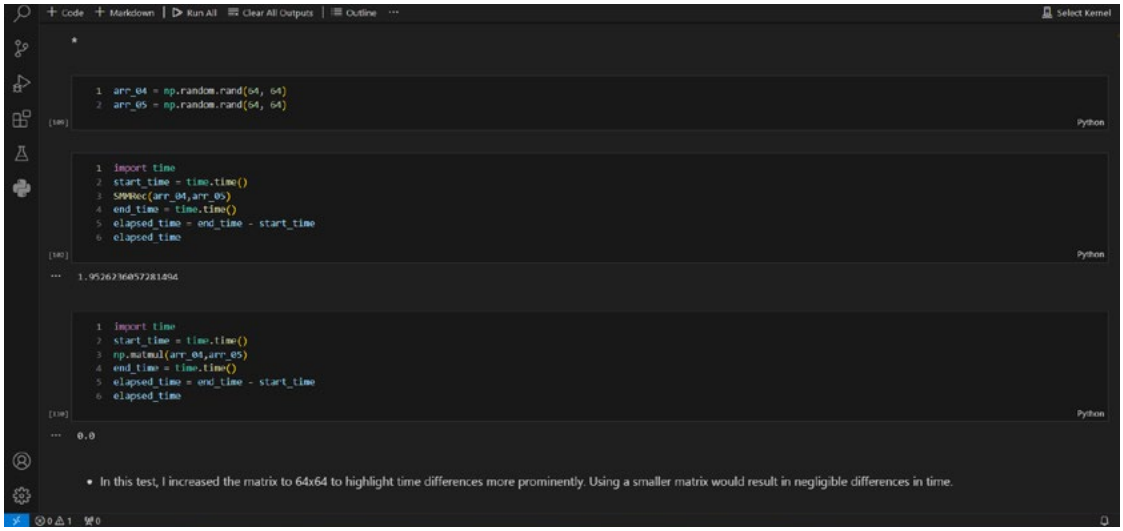


Suggestions 1



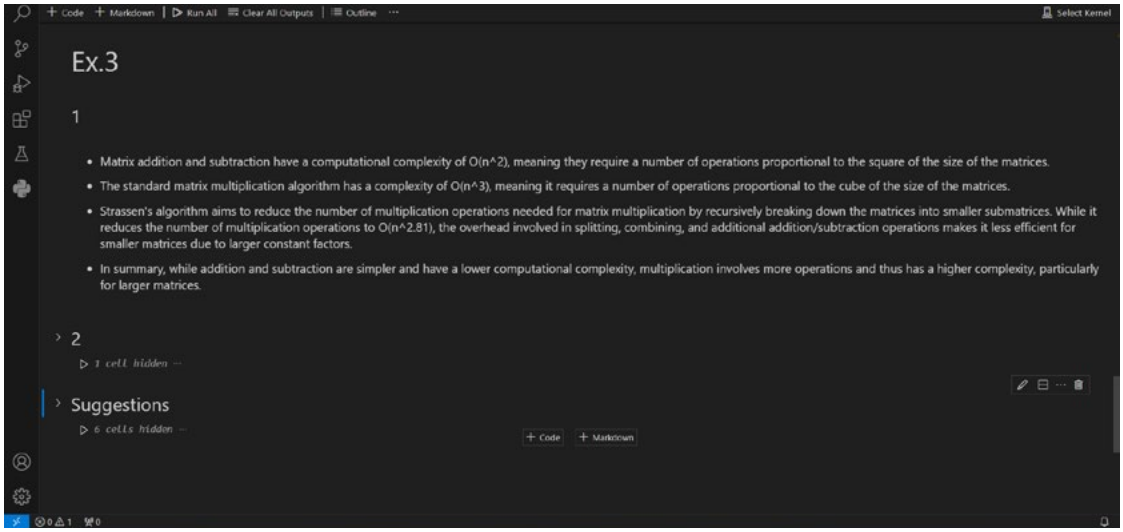
Suggestions 2

Generate 2 random 64x64 matrices. Then import time to count the running time for each function for comparison. In this test, I increased the matrix to 64x64 to highlight time differences more prominently. Using a smaller matrix would result in negligible differences in time. The results show that numpy function takes less time.

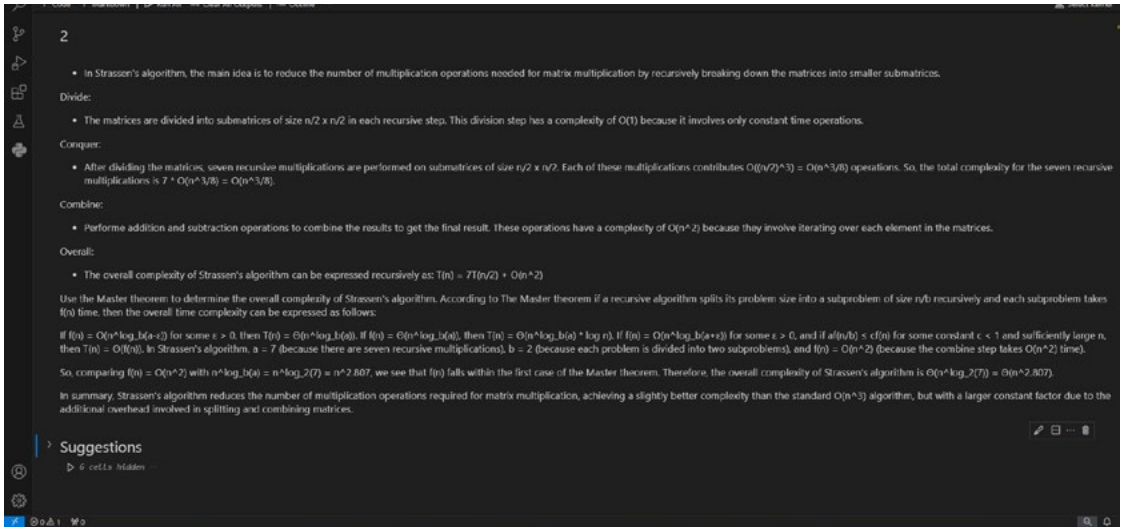


X. EXERCISE THREE

Question 1

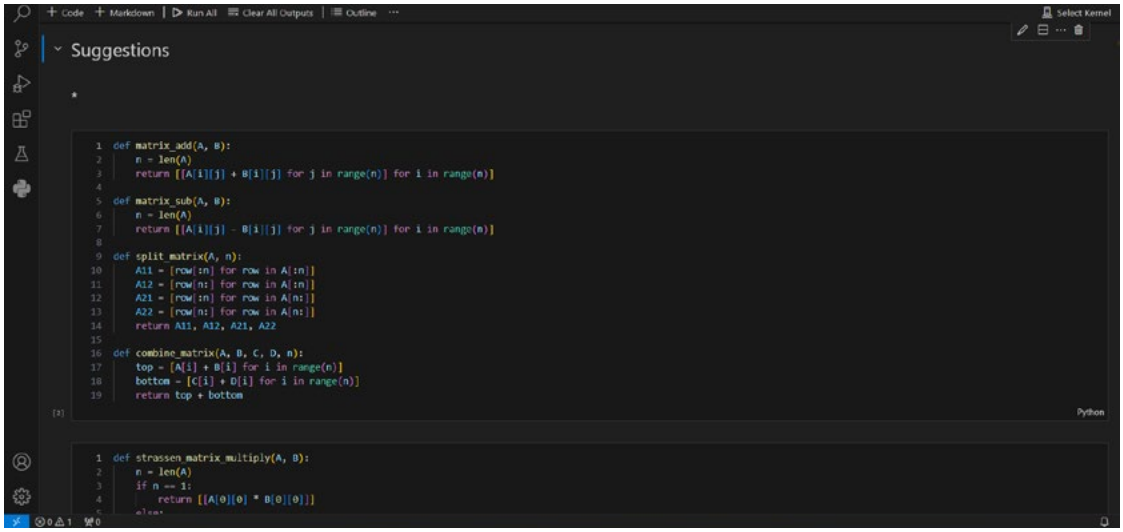


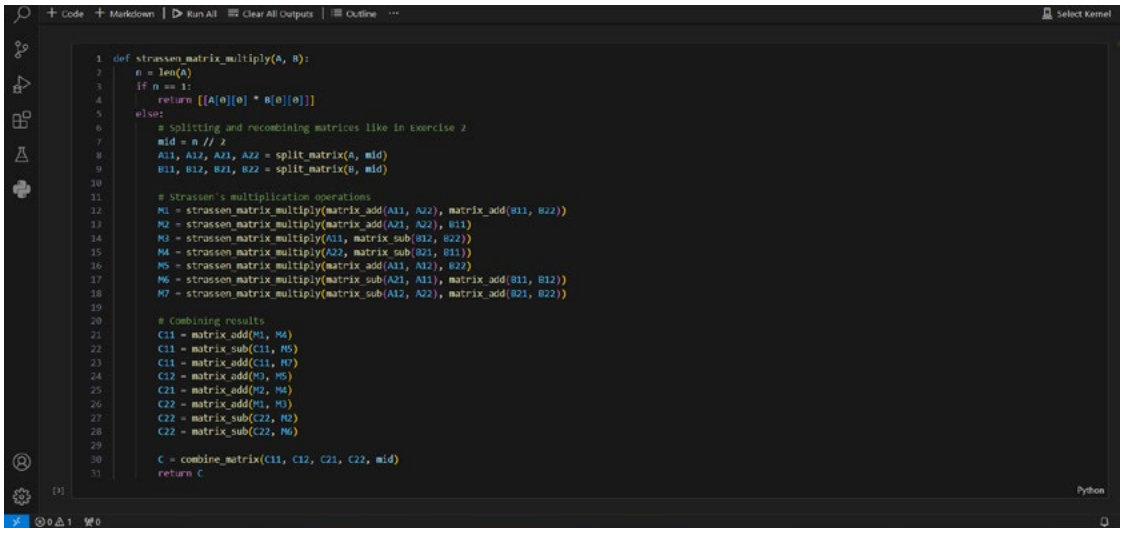
Question 2



Suggestions 1

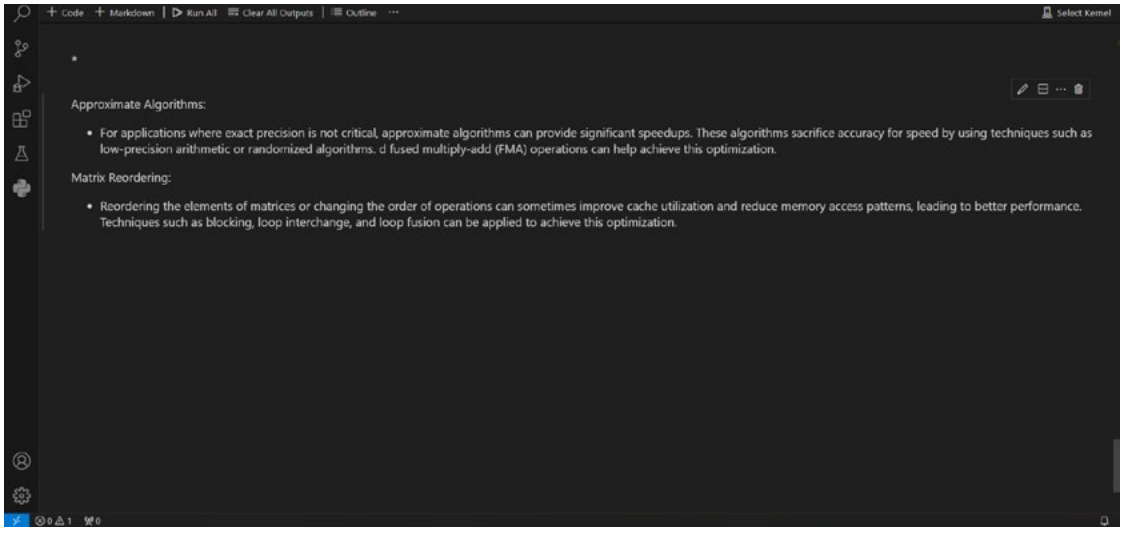
Translate the pseudo code to the notebook.





Suggestions 2

Translate the pseudo code to the notebook.



BIBLIOGRAPHY

Amundson, Jhennifer. (2015). 1983 Porphyrios - Jhennifer A. Amundson, ph.D.. jhenniferamundson. http://jhenniferamundson.net/wp-content/uploads/2015/12/1983-Porphyrios.pdf

Congress for the New Urbanism. (2012, May 30). CNU 20 - heterodoxia architectonica. YouTube. https://www.youtube.com/watch?v=aEdQIYbr9hE

Advanced search. RIBApix. (n.d.). https://www.ribapix.com/

London's Screen Archives. (n.d.). https://www.londonsscreenarchives.org.uk/search-results?q=&filter=theme%2C26%2Chasvideo%2C1&sort=hasvideo#900 dv=false&cid=0&mid=0&vid=0&q=columns&sid=false&isc=true&orderBy=0&pagesize=24&pagenumber=1

RC11 SKILLS FINAL REPORT

Research Cluster 11
Student Number 23188508