**Project Design Phase-II**
**Technology Stack (Architecture & Stack)**

| | |
|---|---|
| Date | 30 October 2023 |
| Team ID | Team ID - 592449 |
| Project Name | Project - Fake/Real Logo detection using Deep Learning |
| Maximum Marks | 4 Marks |

**Technical Architecture:**

The Deliverable include the architectural diagram as below and the information as per the table1 & table 2

**Topic: Fake/Real Logo detection using Deep Learning**

**Reference: 1 ) ML.drawio - draw.io (diagrams.net)**

**2 )**
**https://www.google.com/search?q=technical+diagram+for+deep+learning+models+real+or+fake+logo+detection+steps+&tbm=isch&ved=2ahUKEwiN5qmB3p2CAxVmuScCHdq2CyoQ2-cCegQIABAA&oq=technical+diagram+for+deep+learning+models+real+or+fake+logo+detection+steps+&gs_lcp=CgNpbWcQAzoECCMQJ1CHBljIDWCLGmgAcAB4AIAB4gGIAfULkgEFMC43LjGYAQCgAQGqAQtnd3Mtd2l6LWltZ8ABAQ&sclient=img&ei=ZZw_Zc2fGebynsEP2u2u0AI&bih=714&biw=1536&rlz=1C1RXQR_enIN1034IN1034#imgrc=mxWKMKN3flLNAM**

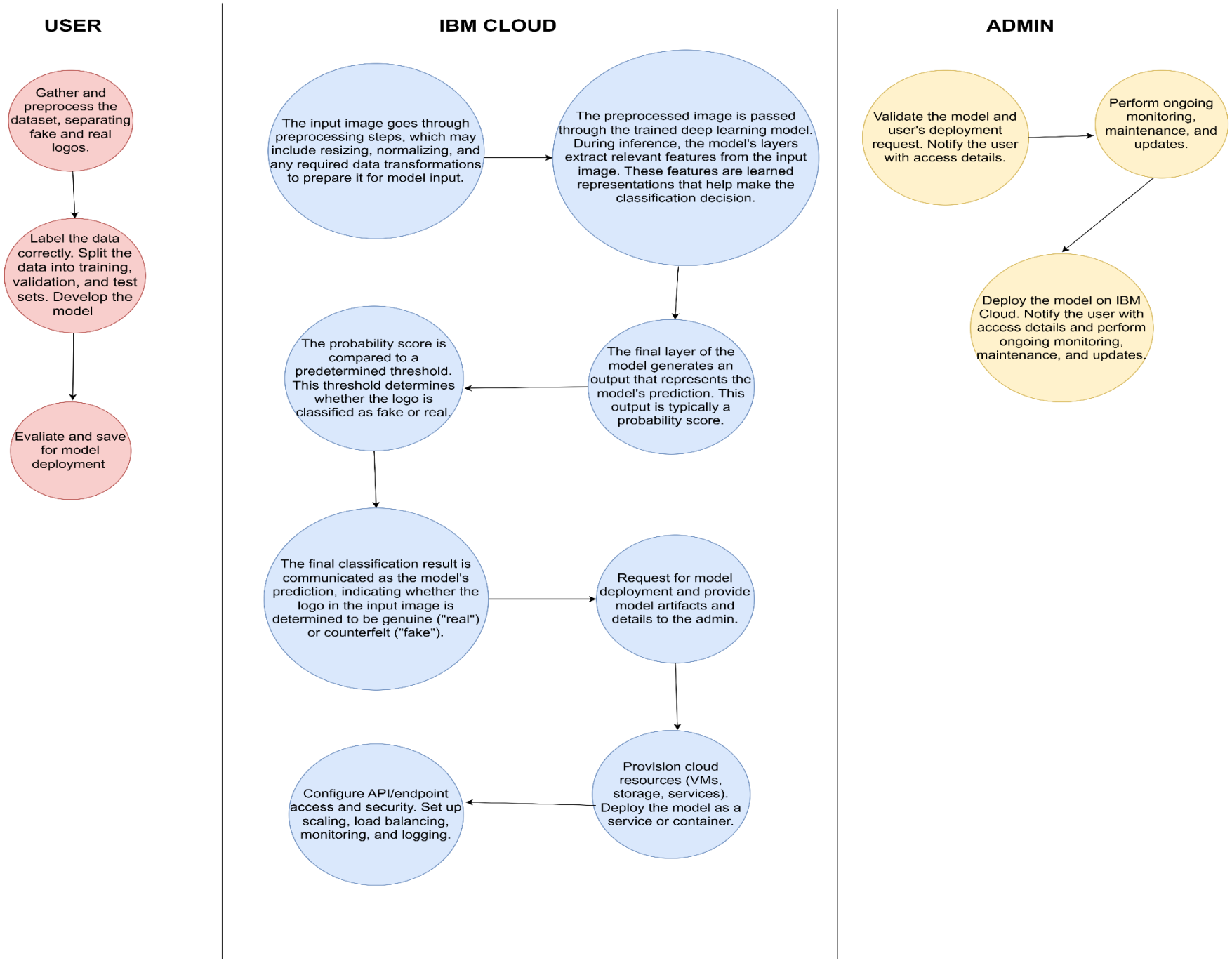
USER
Gather and preprocess the dataset, separating fake and real logos.
Label the data correctly. Split the data into training, validation, and test sets. Develop the model
Evaliate and save for model deployment
IBM CLOUD
The input image goes through preprocessing steps, which may include resizing, normalizing, and any required data transformations to prepare it for model input.
The preprocessed image is passed through the trained deep learning model. During inference, the model's layers extract relevant features from the input image. These features are learned representations that help make the classification decision.
The final layer of the model generates an output that represents the model's prediction. This output is typically a probability score.
The probability score is compared to a predetermined threshold. This threshold determines whether the logo is classified as fake or real.
The final classification result is communicated as the model's prediction, indicating whether the logo in the input image is determined to be genuine ("real") or counterfeit ("fake").
Request for model deployment and provide model artifacts and details to the admin.
Provision cloud resources (VMs, storage, services). Deploy the model as a service or container.
Configure API/endpoint access and security. Set up scaling, load balancing, monitoring, and logging.
ADMIN
Validate the model and user's deployment request. Notify the user with access details.
Perform ongoing monitoring, maintenance, and updates.
Deploy the model on IBM Cloud. Notify the user with access details and perform ongoing monitoring, maintenance, and updates.

**Table-1 : Components & Technologies:**

| S.No | Component | Description | Technology |
|---|---|---|---|
| 1. | User Interface | Provides a user-friendly interface for interacting with the system and submitting logo images for classification. | Web development frameworks (e.g., HTML, CSS, JavaScript, React, Angular) |
| 2. | Application Logic-1 | Handles user requests, manages user authentication, and routes requests to the appropriate services. | Backend application framework (e.g., Node.js, Flask, Django) |
| 3. | Application Logic-2 | Contains business logic responsible for preprocessing and formatting incoming images for model input, as well as handling model predictions and post-processing. | Backend application framework, ML libraries (e.g., Python, TensorFlow) |
| 4. | Application Logic-3 | Manages system scalability, load balancing, and coordinates communication between different components. | Load balancers, orchestration tools (e.g., Kubernetes, Docker Swarm) |
| 5. | Database | Stores and manages data related to user accounts, system configurations, and logs. | Relational databases (e.g., PostgreSQL, MySQL) or NoSQL databases (e.g., MongoDB, Cassandra) |
| 6. | Cloud Database | Provides scalable and highly available storage for system data, ensuring data integrity and reliability. | IBM Db2, IBM Cloud Databases, or cloud-native database solutions |
| 7. | File Storage | Stores and manages images, model weights, and other files, making them accessible to various system components. | Cloud Object Storage (e.g., IBM Cloud Object Storage, Amazon S3) |
| 8. | External API-1 | Interfaces with external services or data sources for supplementary information (e.g., logo reference data). | HTTP requests to external APIs |
| 9. | External API-2 | Integrates with additional external services, such as geolocation or third-party libraries for specific functionality. | HTTP requests to external APIs |

| | | | |
|---|---|---|---|
| 10. | Machine Learning Model | Performs logo detection by running inference on input images. The model uses deep learning techniques to differentiate between fake and real logos. | Deep learning frameworks (e.g., TensorFlow) |
| 11. | Infrastructure (Server / Cloud) | Provides the computing resources, such as virtual machines, containers, or cloud services, necessary for hosting and scaling the entire system. | IBM Cloud services, virtual machines, or container orchestration platforms (e.g., IBM Kubernetes Service) |

**Table-2: Application Characteristics:**

| S.No. | Characteristic | Description | Technology |
|---|---|---|---|
| 1. | Open-Source Frameworks | Leveraging open-source deep learning frameworks for building, training, and deploying the machine learning model. | - TensorFlow for deep learning model development. - Scikit-learn for pre-processing and data analysis. |
| 2. | Security Implementations | Incorporating security measures to protect data, user interactions, and system components, ensuring data privacy and system integrity. | - Secure communication protocols (HTTPS).<br>- Authentication and authorization mechanisms.<br>- Data encryption at rest and in transit. |
| 3. | Scalable Architecture | Designing the system to handle varying workloads and adapt to changing demands by employing a scalable and flexible architecture. | - Container orchestration platforms (e.g., Kubernetes) for auto-scaling. - Load balancers to distribute traffic.<br>- Serverless computing (e.g., IBM Cloud Functions) for event-driven scalability. |
| 4. | Availability | Ensuring high availability by minimizing downtime, fault tolerance, and redundancy to guarantee that the system remains accessible and responsive. | - Multi-region deployment for redundancy.<br>- Automated failover mechanisms. - Monitoring and alerting to address issues promptly. |

| | | | |
|---|---|---|---|
| 5. | Performance | Optimizing system performance by using efficient algorithms, reducing latency, and leveraging hardware acceleration for faster model inference. | - Hardware acceleration (e.g., GPUs, TPUs) for deep learning inference.<br>- Profiling and optimization of the deep learning model.<br>- Caching mechanisms for frequently accessed data. |

**References:**


1) **https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/**
2) **https://www.ibm.com/cloud/architecture**
3) **https://aws.amazon.com/architecture**
4) **https://www.lucidchart.com/blog/how-to-draw-architectural-diagrams**