Tech Feasibility

Team: INSIGHT
Sponsor: Mike Taylor
Faculty Mentor: Scott Larocca
Team Members:
Joshua VanderMeer, Michael Vertin,
Aidan Hebert, Forrest Hartley

Date: 10/25/2024

Table of Contents

1. Introduction

The Cline Library offers a plethora of valuable resources for researchers, students, faculty, and curious individuals alike. One noteworthy resource that the Cline Library hosts is its special collections and archives (SCA). The special collections and archives are responsible for collecting, organizing, and preserving important images that can inform our historical perspective and help to provide insight into past events.

Currently, more than 100,000 items have been digitized and made available through the library's SCA via a content management system by the name of CONTENTdm. CONTENTdm currently manages all of the library's collections in a secure cloud-based preservations archive. Additionally, CONTENTdm also handles all filtered searches throughout the database, giving users the ability to narrow down their search with a range of varied criteria. This more traditional method of searching can certainly be effective in many instances, but it isn't without its disadvantages. Because items within the collection are manually digitized and labeled, queries oftentimes aren't exhaustive. It's unrealistic to categorize every digitized photo within the system according to all of its characteristics at the scale that the Cline library is operating at.

Given that this vast collection of resources is only as valuable as the means by which it can be accessed, effective searching methods are a must. For such a large collection of items that will only grow in size going forward, it's most important that items of interest can easily be accessed with consistency and precision.

This is where machine learning can come in. With the power of AI, we can use pre-trained models to sift through images by identifying characteristics specific to each image in order to retrieve more accurate and precise search results. This categorization method can work in conjunction with pre-existing metadata to provide even more relevant images.

Not only do we hope to improve this process, but additionally we also plan on implementing a search by image feature to allow for queries based on images provided by the user. This enhancement addresses a common challenge: users who are uncertain of the best keywords or categorizations for their queries. In such situations, building upon a pre-existing image to find others with similar themes or classifications makes sense and can greatly improve the user experience.

Once an image has been inputted, image recognition technologies can then retrieve prevalent characteristics from that image and compare them against the other digitized images stored in the database to fetch the most similar images based on their perceived characteristics.

As a whole, we envision a responsive and intuitive web-based application which can take user-inputted images and return a grid of the most similar images back to the user. Similarity would be determined by comparing calculated embeddings of the uploaded images with that of the other images in the database using some form of similarity measure, such as cosine similarity. From this point, users would be able to select an image from the displayed grid and watch as the app responsively updates the grid to correlate with the selected image, which would then function as the new "seed" image. As users traverse from image to image, a history of past images will be kept to enable backtracking.

With this in mind, our end goal is to significantly enhance the accessibility and usability of the Cline Library's digital collections by providing valuable tools that can benefit the user base regularly. We hope to not only improve search precision for the library's image database but also to further the library's mission of preserving and sharing the rich narratives housed within the Special Collections and Archives.

Now having covered a rough outline of the necessity for image searching within the Cline Library's SCA, and our general approach to the problem, we can turn our attention to the technological challenges that we anticipate going forward. With these potential technological challenges in mind, we will be better prepared for any setbacks that may occur, and more equipped to avoid dead-end approaches altogether.

2. Technological Challenges

- We will need to effectively communicate between our web app and our vector database.
- We will need to communicate between our web app and the CONTENTdm platform.
- We will need to save vector embeddings for future comparisons.
- We will need to carry out all vector embedding comparisons within an acceptable window of time to maintain a responsive front-end experience.

- We will need to find the best similarity measure for comparisons within our vector database.
- We will need to create a web app that is presentable, and acceptable for both the Cline Library and Northern Arizona University
- We will need to create scalable features that can be maintained as the database grows in size.
- We will need to ensure that user-uploaded images are securely transmitted and stored.
- We will need to include support for various image formats.
- We will need to handle preprocessing tasks for images such as resizing, normalization and orientation correction.
- We will need to create a web app that provides a consistent user experience across devices and browsers.
- We will need to smoothly integrate our technologies into the existing library's system.
- We will need to implement error handling mechanisms for graceful exception handling.
- We will need to maintain the security of the collections and archives by guarding against malicious attacks.
- We will need to implement diagnostic and performance related tools to gather metrics.
- We will need to design our web app with future proofing in mind.
- We will need to ensure all components of our tech stack are compatible and can be updated independently if necessary.
- We will need to implement accessibility features to accommodate all users.
- We will need to make our web-app intuitive and provide support/documentation for features that are less self-explanatory.

3. Technology Analysis

The challenge we have to develop is a functioning application for students and faculty to have access to the NAU library archives through image similarity searching to find the images they are looking for. The application will also be able to allow manual searching still as it is currently set up. Not only will our application have to be able to find similar images to the user input but it will also display these results in a neat and usable way for the user. We also have to consider the possibility of an image that has nothing similar in the whole collection and what its near possible match would be. All this together forms the main problem which our solution will be developed to solve.

With this overall picture, we have a large amount of functionality and tools to consider for this project. We will need to start with a vector database that will store the archived images in vector format for future comparisons. With this, we will need to develop a tool that will take a photo from the user and read, create, embed, and vectorize the image. From this vector of the image, we will have a tool capable of performing vector cosine calculations to find the top nine most relevant images that are similar to the user's image. From here we will need to have an API locate the images that have been selected from the contentDM database, store the images and send them to the front end to be displayed. From here there will be a website built in react, express, and node to be able to process this incoming data and display it functionally for the user to interact with. With all of this, we will be able to have a functioning application that users will be able to use to find images that fit their needs.

3.1 Technology Analysis - Front End/Web Design

3.1.1 Introduction

The development of an image similarity search application for the Cline Libraries Special Collections and Archives (SCA) presents a few key technological challenges, one of these being building a responsive, interactive, and scalable web application for the target user base. The goal is to allow users to upload images and find similar images in the collection. This would allow for a visual result that effectively renders similar images. The chosen stack must facilitate real-time image uploads, efficient similarity tools, and a clear presentation of results. This project requires a solution that integrates React for the front end and Node.js and Express for backend integration with the machine learning algorithms and tools developed for embeddings and vector calculations.

3.1.2 Desired Characteristics

- ***Interactivity and Responsiveness (40%)***: the user interface should provide rapid visual feedback when an image is uploaded. This allows users to interact with the system seamlessly. Users should experience a smooth and quick transition from the different views of the website. These views would include uploading an image, processing, and viewing results. An interactive UI would also include a responsive and dynamic grid of the results. This grid would include image thumbnails generated by CONTENTdm.
- ***Scalability (25%):*** This web application must be able to handle large amounts of data from the growing dataset of over 100,000 digitized images. It needs to be able to handle simultaneous user requests and have minimal delays for users. Must have a scalable architecture to optimize backend logic to ensure the search quality and image similarity computation effectiveness.
- ***Clear Visual Representation (15%):*** Results should be displayed in a clean, organized, and intuitive manner. The application needs a visually appealing design that groups similar images cohesively and provides contextual information to aid the user's understanding of the search results.
- ***Flexible Back-End Support (10%):*** The back end must be designed to efficiently handle multiple concurrent user interactions, including image uploads, embedding calculations, and similarity searches. It should be robust and scalable allowing for smooth expansion as the dataset grows or as new features are integrated. This ensures consistent performance and reliability.
- ***Maintainable and Modular Codebase (10%):*** The project should implement a clear separation of concerns between the front end and back end following a modular architecture. This approach will facilitate future modifications, debugging, and

enhancements, enabling the development team to independently update and maintain different components without affecting the entire system.

3.1.3 Alternatives

- ***Front End:*** Some options for front-end development are Angular, Vue.js, and React. Angular provides a comprehensive Model-View-Controller (MVC) framework and built-in features, while Vue.js stands out for its simplicity and ease of use when integrating with existing frameworks and projects. React was selected due to its flexibility, vast ecosystem, and strong support for building dynamic and interactive UIs.
- ***Back End:*** One option for back-end development is Django, a Python-based framework, which is a strong option due to its high-level abstraction and built-in tools for rapid development. Similarly, Ruby on Rails is a developer-friendly environment with conventions that are efficient and support a quick development cycle. Node.js combined with Express was chosen for its non-blocking architecture flexibility and lightweight framework. This aligns well with the need for real-time data handling and scalable API routing needed for this project.
- ***Full Stack Frameworks:*** The MEAN stack, which includes MongoDB, Express, Angular, and Node.js, is a popular full-stack approach that emphasizes consistency with JavaScript on both the client and server sides. Alternatively, the MERN stack, using React instead of Angular, is another robust choice for building dynamic web applications. The combination of React, Node.js, and Express was selected to achieve consistency in language and enable efficient development and maintainability.

3.1.4 Analysis

React.js

- ***Interactivity and Responsiveness:*** Node.js's non-blocking architecture is efficient for handling concurrent requests, enhancing backend responsiveness, while Express simplifies request handling for a seamless front-to-back experience.
  Score: 8

- ***Scalability:*** Node.js's asynchronous nature and scalability are well-suited for handling high user load and image processing tasks, particularly when integrated with a load-balanced environment.
  Score: 8
- ***Clear Visual Representation:*** While Node.js and Express focus on backend processes, their ability to deliver data quickly supports clear frontend rendering. The backend performance indirectly aids in maintaining clean visuals.
  Score: 7
- ***Flexible Back-End Support:*** Node.js and Express provide strong support for APIs and image processing tasks, offering the flexibility needed for complex back-end tasks like image uploads and embedding calculations.
  Score: 9
- ***Maintainable and Modular Codebase:*** Node.js and Express are known for simple, modular routing and middleware that encourage maintainable code, allowing developers to manage separate features independently.
  Score: 8

Node.js and Express

- ***Interactivity and Responsiveness:*** React's virtual DOM and component-based design allow for highly responsive and interactive UIs. The virtual DOM will handle UI updates efficiently, making it suitable for seamless image upload and display.
  Score: 9
- ***Scalability:*** While React can handle complex UIs, its scalability relies on the backend and state management libraries. React itself may need additional tools like Redux or server-side rendering to fully scale for high-concurrency usage.
  Score: 7
- ***Clear Visual Representation:*** React's component-based structure promotes reusability, allowing for a cohesive, intuitive UI. Customization is straightforward, and frameworks like Material-UI further enhance visual consistency.
  Score: 8

- ***Flexible Back-End Support:*** React is a frontend library and depends on backend APIs. While it works well with robust backends, React alone doesn't directly contribute to backend flexibility.
  Score: 5
- ***Maintainable and Modular Codebase:*** React's component model makes it easy to organize and manage code, which is highly maintainable. Components can be reused across the application, promoting a modular structure.
  Score: 9

3.1.5 Chosen Approach

For the development of the image similarity search web application, React will be used for the front end, and Node.js with Express for the back end. This decision leverages React's capabilities to create dynamic and interactive user interfaces. While Node.js and Express facilitate efficient image processing with a robust server-side logical design. Together these two tools will ensure a responsive and intuitive web-based application that fits the needs of this project.

Web Stack	Responsiveness	Scalability	Visual	Flexible	Maintainable	Total
React	**9**	**7**	**8**	**5**	**9**	**7.7**
Express & Node	**8**	**8**	**7**	**9**	**8**	**8**

3.1.6 Proving Feasibility

Moving forward from this initial analysis further steps will be taken to validate the selected solution above. This will ensure that the solution is an effective choice for this image similarity tool. To demonstrate the feasibility of using React, Node.js, and Express the team will develop a small technology demo. This demo will show off the user interface portion of the product to show its design. Another demo will show off the image upload capabilities and embedding generation process. Finally leading to a full product working with the SCA image database. These demos will continue to prove that this stack is a feasible solution to this web application problem

3.2 Technology Analysis - Hosting Database

3.2.1 Introduction

The development of this image similarity application requires a robust database hosting solution. Given the large volume of digitized images that require the need of storing and retrieving embeddings of these images. This paired with the need to store this data in a vector database choosing the right hosting platform is crucial. The database must support fast querying, handle large-scale data, and integrate seamlessly with machine learning algorithms and image similarity measures. AWS and DigitalOcean are the two potential contenders for this, each with unique benefits and trade-offs.

3.2.2 Desired Characteristics

- ***Compatibility (50%):*** The database hosting solution must be compatible with the other project components, including node.js, express, the chosen database structure, and the machine learning algorithms used for calculations. Compatibility with machine learning frameworks such as TensorFlow and Hugging Face is essential to support the project's core functionality and image search functionality.
- ***Scalability (20%):*** With an ever-growing number of images and a staggering 100,000 already digitized images the host solution must scale seamlessly to accommodate this. With the increasing data volume and simultaneous user requests. Horizontal scaling capabilities, such as auto-scaling database clusters would be an ideal solution to support potential growth.
- ***Flexibility (15%):*** The platform chosen needs to be flexible enough to support the ever-changing needs of the project. Such as the ability to adjust the size of database capacity, and machine learning integration. This as well as a flexible and robust server management configuration helps with optimizing both cost and performance.
- ***Easy to use/learn (15%):*** The platform should provide an intuitive interface and well-documented tools to facilitate easy management of database services. Clear guidelines for configuring instances, monitoring resources, and optimizing database performance are critical for minimizing overhead on the development team.

3.2.3 Alternatives

- ***Database Hosting:*** Amazon Web Services (AWS) offers a comprehensive range of services including scalable databases. AWS also provides easy integration with models and machine learning for image embeddings. DigitalOcean another hosting platform provides a more cost-effective solution but is aimed more towards smaller projects. DigitalOcean would be able to host the entire application as it has support for the front-end stack. DigitalOceans ecosystem is less extensive than AWS and may face challenges with the scalability and security of this large-scale project.

3.2.4 Analysis

Amazon Web Services (AWS)

- ***Compatibility**:* AWS offers extensive compatibility with machine learning frameworks like TensorFlow and Hugging Face, which is critical for this project. Its support for Node.js, Express, and various database options (RDS, DynamoDB) ensures seamless integration with the project stack.
  Score: 9
- ***Scalability**:* AWS excels in scalability with features like auto-scaling and managed clusters, suitable for accommodating a growing dataset and user base. The infrastructure is designed for large-scale projects, ensuring smooth horizontal scaling as needed.
  Score: 9
- ***Flexibility**:* AWS provides a flexible environment, allowing adjustments to database capacity, instance types, and machine learning configurations. This flexibility supports both cost optimization and performance tuning.
  Score: 8
- ***Easy to Use/Learn**:* AWS has a steep learning curve due to its complexity and vast feature set. However, it provides extensive documentation and robust support, which can help mitigate the challenges.
  Score: 6

DigitalOcean

- ***Compatibility**:* DigitalOcean supports Node.js and managed databases, making it compatible with the project's needs. However, it has limited direct integration with advanced machine learning frameworks, which might necessitate additional configuration.
  Score: 6
- ***Scalability**:* DigitalOcean can handle moderate scaling, but lacks advanced auto-scaling features and horizontal scaling capabilities found in AWS. This limitation could impact its ability to manage the growth demands of a large, data-intensive project.
  Score: 5
- ***Flexibility**:* DigitalOcean offers adjustable database capacities and has a relatively straightforward server management approach, which can help manage cost and performance within a smaller scope but lacks the robustness of AWS.
  Score: 6
- ***Easy to Use/Learn**:* DigitalOcean has an intuitive interface and clear documentation, making it easier to learn and use for simpler configurations, especially for smaller projects.
  Score: 8

3.2.5 Chosen Approach

After evaluating the requirements of the image similarity search application and considering the trade-offs, AWS is chosen as the preferred hosting solution. Its scalability, comprehensive range of services, and advanced security features align with the needs of handling large datasets and secure storage. AWS's extensive service offerings also facilitate easy integration of machine learning workflows, making it the best option for efficiently hosting and managing the database for this project.

Hosting	Compatibility	Scalability	Flexibility	Easy to Use	Total
AWS	**9**	**9**	**8**	**6**	**8.3**
DigitalOcean	6	5	6	8	6.1

3.2.6 Proving Feasibility

Several demos will be developed to validate the choice of AWS as the hosting platform. For example, we will show off the successful hosting of the embedding and vector data on an AWS hosted database. It has been proven that AWS is a popular hosting platform for many tech products in the industry as they are one of the leading cloud hosting providers.

3.3 Technology Analysis - Vector and Machine Learning Libraries

3.3.1 Introduction

Currently, there are many tools for machine learning. This section will analyze libraries designed to train models and search through information in vector databases.

3.3.2 Desired Characteristics

Several tools include both machine learning and vector database searching. For this reason, we decided to combine these functionalities. The chosen approach may consist of multiple alternatives to ensure both machine learning and vector analysis are represented well. The scores provided for machine learning and searching will not impact the alternative's total score because they do not represent the quality of the alternative.

- ***Compatibility (50%)***: Compatibility is a tool's ability to be efficiently integrated into other tools. Training requires direct access to the sources, and vector searching requires references to embeddings that correspond to sources in existing other locations. For these reasons, we chose compatibility as the highest weighted characteristic.
- ***Simplicity (30%)***: We have little experience with machine learning and interacting with large databases. Simple tools allow us to implement these algorithms without needing to understand the mathematical theory behind modeling, training, and searching. This allows us to focus our effort into other areas of the project.

- ***Controllability (20%)***: Controllability is our ability to control the system's outcome. A tool with low control cannot be adjusted if its outcome differs from our expectations. A tool with high control can be modified to accurately model our expectations. Having a functional system is more valuable to us than having complete control, so we chose this as our lowest weighted characteristic.
- ***Machine Learning/Vectors***: Ideally, one tool will provide everything we need for both machine learning and vector analysis. However, we may need multiple tools to be compatible with other parts of our system. Additionally, using multiple compatible tools will allow us to combine the strengths of each system. This characteristic describes whether the tool is designed for machine learning, vector analysis, or both.

3.3.3 Alternatives

- **Tensorflow** is a tool for developing deep-learning models. It allows developers to create, train, and fine-tune models to data sets. Tensorflow provides libraries for high control over the learning architecture without specifying the calculations used during training. Developers also have complete control over how vigorously models are trained.
- **HuggingFace** is an open source platform specifically designed for AI tools, such as learning models, data sets, and AI applications. The large number of contributors makes HuggingFace a popular choice to quickly integrate a basic AI component.
- **ScaNN** is an optimized nearest neighbor search tool developed by Google. ScaNN is designed to maximize similarity searching performance in vector databases with billions of sources. ScaNN provides simple functions for general searching, and allows developers to modify parameters and searching algorithms for fine-tuning to specific use cases.
- **Milvus** is a vector database tool dedicated to fast database access. Milvus was specifically designed for handling unstructured data and embeddings for machine learning. Milvus supports many built-in indexing types and search algorithms to provide developers with easy database access without sacrificing efficiency.
- **Pytorch** is a detail oriented approach to machine learning. This provides functions for the fundamental calculations in machine modeling and training, and leaves the rest of the

design to the developers. Consequently, pytorch provides high control over the learning process, and requires a deep understanding of machine learning.

3.3.4 Analysis

Tensorflow

- ***Compatibility:*** Tensorflow provides an API for importing data from a database, and allows data to be imported in any directory. Tensorflow also offers deployment options for mobile devices and browsers. ScaNN has a feature specifically designed to work with Tensorflow.
  Score: 9
- ***Simplicity:*** Tensorflow has separate libraries for each step in constructing machine learning models. This keeps the process organized and intuitive, but requires developers to have a deep understanding of the model's structure. Additionally, developers are responsible for iteratively training the model, contributing to the program's complexity.
  Score: 6
- ***Controllability:*** Developers have high control over the basic characteristics of each layer and the learning process. Additionally, Tensorflow gives developers high control over how much the machine's training process. This allows developers to determine how much training it has on any combination of data sets.
  Score: 9

HuggingFace

- ***Compatibility:*** HuggingFace has been implemented in many large businesses including Microsoft, AWS, and Google. There are also guides to accessing data from other services such as Milvus.
  Score: 7
- ***Simplicity:*** HuggingFace provides hundreds of thousands of pre-trained models, data sets, and code spaces made by users, which can be modified as needed. The libraries it

provides are very high-level, requiring minimal experience to implement into any system.
Score: 10

- ***Controllability:*** Due to HuggingFace’s tendency toward high-level machine representation, most of the models provide little control over the machine’s learning style.
Score: 2

ScaNN

- ***Compatibility:*** ScaNN has APIs for Tensorflow and Python APIs, and is supported by Linux.
Score: 8
- ***Simplicity:*** ScaNN provides basic search tools for determining how close two embeddings are semantically. It also provides functions for more complex purposes with consistent formatting. Because ScaNN isolates itself to vector arithmetic, there are a small number of concepts to become familiar with.
Score: 9
- ***Controllability:*** The functionality that ScaNN provides is very granular, giving developers high control over how functions can be organized and combined. Due to ScaNN’s small focus area, the tools it provides make up a significant portion of what is possible.
Score: 8

Milvus

- ***Compatibility:*** The process for importing data requires a specific format called ‘Milvus bucket’ through MinIO. Then, an import job is needed to access the data in the program. There are built-in functions for storing and retrieving models generated with Milvus.
Score: 5
- ***Simplicity:*** The built-in search algorithms allow developers to easily access database information without understanding the algorithms’ process. For training, most of the process is automated.
Score: 9

- ***Controllability:*** Milvus provides a great deal of control over data searching. However, developers do not have direct access to embeddings in the database due to how long it can take to iterate over everything.
  Score: 8

pytorch

- ***Compatibility:*** Pytorch uses simple data structures, making it easy to implement into any python library that supports low-level modifications.
  Score: 8
- ***Simplicity:*** Pytorch requires a lot of knowledge of how machine models work, making this library very challenging for inexperienced developers. It requires a significant amount of the algorithm to be designed by its users.
  Score: 1
- ***Controllability:*** Pytorch leaves the machine learning process' functionality to the developers, resulting in nearly full control over the data.
  Score: 9

3.3.5 Chosen Approach

For machine learning and vector database analysis, we chose a tool designed for each of the following sections: machine learning, vector analysis, both.
These options excelled in the characteristics that we selected. Additionally, they have specific integration options for each other, which furthers their compatibility value.

	Compatibility (.5)	Simplicity (.3)	Controllability (.2)	**Total (1)**	Learning/Vectors
Tensorflow	**9**	**6**	**9**	**8.1**	**Learning**
HuggingFace	7	10	2	6.9	Learning
ScaNN	**8**	**9**	**8**	**8.3**	**Vectors**
Milvus	**5**	**9**	**8**	**6.8**	**Learning+Vectors**

pytorch	8	1	9	6.1	Learning+Vectors

3.3.6 Proving Feasibility

Tensorflow, ScaNN, and Milvus have different strengths that we can combine to maximize our system's value. ScaNN pairs well with Tensorflow, and Tensorflow provides an API to help with connecting to the database.

3.4 Technology Analysis - Machine Learning Model

3.4.1 Introduction

This section will determine the type of model the machine will use to interpret images. This will determine the quality of the search results, which is a high priority for the project.

3.4.2 Desired Characteristics

- ***Flexibility (45%):*** Our project will initialize the ability to search through images on the SCA. While we know the current expectations of this system, we cannot predict the additions and modifications made to this system in the future. For this reason, we chose flexibility as the heaviest-weighted characteristic.
- ***Scalability (35%):*** The SCA already contains tens of thousands of images, and will continue to grow in the future. As we receive more sources of information, the ideal system will incorporate that information into its memory. The accuracy of searches is another high priority in the system.
- ***Controllability (20%):*** Controllability is our ability to control the machine's decisions. A model with low control may not align with our concept for similar and different images. A model with high control, however, allows developers to fine-tune the model's interpretations as needed.

3.4.3 Alternatives

- **CNN**s (Convolutional Neural Network) are a model for classifying images. These models identify the position of features in an image and use those features to determine which object(s) it represents. The output is a vector, where each index corresponds to an object type, and each value represents how well the features match expectations. CNNs are commonly used for object detection and image classification.
- **ViT**s (Vision Transformer) are used for pairing images of similar meaning. These models aim to maximize the connotation that can be stored in an embedding. Consequently, the significance of an embedding cannot be directly interpreted by humans. Instead, the output can be indirectly interpreted by searching for other nearby embeddings, which are associated with images, texts, or other media with similar connotations. Transformers are commonly used for searching and generation.

3.4.4 Analysis

Vision Transformers (ViT)

- ***Flexibility:*** Transformers are very flexible because they store compressed interpretations instead of features. This means ViTs can be trained to share the same embedding space as different forms of media, allowing searches for text using images and searches for images using text.
  Score: 8
- ***Scalability:*** There is no limit to how many objects the model can recognize. As the model receives information to train on, it can continue adjusting its semantics to fit the new information.
  Score: 9
- ***Controllability:*** Transformers are difficult to control because no component in a transformer has an interpretable output. The only control that developers have is what information the AI has access to.
  Score: 2

Convolutional Neural Network (CNNs)

- ***Flexibility:*** CNNs produce classification embeddings, which restricts how much information can be extracted from the image. Similar to ViTs, other models can be trained to create a similar embedding space.
  Score: 7
- ***Scalability:*** The types of objects that CNNs can detect are predetermined and limited to the size of their output embedding. The CNN must be retrained in order to classify other types of objects. Because each index corresponds to one object, the CNN needs to calculate its similarity to every type of object that it can identify.
  Score: 5
- ***Controllability:*** Controlling classification is easy in CNNs due to its human-readable output. Additionally, CNNs are designed to find the position of features, so developers are able to identify quantities and positional relationships between objects.
  Score: 9

3.4.5 Chosen Approach

For the learning model, we chose ViT. While transformers are hard to control, their semantic-driven representation makes ViTs incredibly flexible and scalable. ViT's are compatible with other data types, and their ability to continually adapt ensures a longer lifespan.

Learning Model	Flexibility (.45)	Scalability (.35)	Controllability (.2)	**Total (1.0)**
ViT	**8**	**9**	**2**	**7.15**
CNN	7	5	9	6.7

3.4.6 Proving Feasibility

Transformers are frequently implemented for learning models. From researching transformers this semester, we already have a basic understanding of how transformers function, and there are many guides for implementing them into a variety of applications.

4. Technological Integration

4.1 Introduction

For the successful completion and development of the image similarity search application for Cline Libraries Special Collections and Archives (SCA), we must take each of our previously discussed approaches to individual issues, and compile them into a comprehensive application to facilitate interactions with the SCA archives. This will require the integration of numerous technologies that have been outlined above including both front and backend systems.

4.2 Frontend Website Integration

For proper and seamless integration with other systems outlined in the prior section of this document, we have chosen to go the route of a React plus nodeJS for our main front-end implementation. This route will ensure that we have a dynamic interface that will not only provide a solid foundation for the project but will also provide the ability to seamlessly scale with the ever growing dataset this project will work with. This interface scalability is crucial for the longevity and user experience of this tool. Additionally, NodeJS and Express will greatly benefit the integration of our frontend system with our backend system. Providing us with a seamless middleware that aims to bridge the gap between user requests and our database. Additionally, express and NodeJS will provide a consistent environment for any and all maintenance and updates.

4.3 AWS Backend Integration

To support and host our previously described frontend, we will require a flexible hosting platform that will support seamless integration between our frontend interface, machine learning model, and database. AWS has already been proven as a viable solution for systems that handle large amounts of embedding and vector data. This will allow for a seamless integration between the frontend and backend. Additionally, the scalable infrastructure AWS provides will benefit our use of Vision Transformers as this model is not bound by anything other than the amount of data it is given. This hosting platform will provide a scalable and well integrated environment to house a system of this nature.

5. Conclusion

5.1 Technology Summary

In summary, our system aims to be flexible and scalable to allow for future expansion and maintenance. To achieve this goal we have devised a set of technologies that will be best suited for this system. This system begins with our frontend implementation, for which we have chosen React. This is due to its flexibility and responsiveness to user interfaces. This choice will ensure a seamless user experience when paired with the backend which will be powered by NodeJS and Express. This backend choice will provide a robust framework to handle any API requests and facilitate efficient image processing and embedding calculation. When considering our hosting options, we found that AWS served to support a system of this nature the best while providing the best in house tools for dealing with and scaling large vector data sets. This hosting platform will additionally help facilitate interactions between our machine learning model of choice ViT, and various machine learning libraries specified above. Vision transformers will additionally support the scalable nature of this project as it will not be bound by anything other than the amount of data the model has access to. These separate systems will serve as the best solutions to help bring this project to fruition and satisfy the scaling and longevity needs this project requires.

5.2 Importance of the Issue & Conclusion

In conclusion, to achieve the described system for the Cline Library Special Collections (SCA) we must tackle this multifaceted problem with care and regard for each individual system necessary for the development and future maintenance of this system. Additionally, this system aims to serve as the future search tool for all SCA collections going forward and should be built with this idea in mind. We hope our careful selection of technologies and methods reflects our commitment to creating a sustainable and adaptable system to assist with the issue of searching efficiently through SCA's massive media collection while providing a seamless user experience for future interactions with this collection.