MODEL SELECTION

~ Kevin Shah

1. PREDICTING BACTERIAL PROPERTY

For the task of predicting bacterial properties given a dataset with 20,000 rows and 150 columns, where each column describes a property of the bacteria, and considering the constraint of limited computational resources (no access to a GPU cluster), I would suggest using a Gradient Boosting algorithm, specifically LightGBM.

I. **LightGBM** is a gradient boosting framework that offers high accuracy, efficiency, and handles large-scale datasets well. It is known for its speed and memory efficiency, making it suitable for training on a potato computer without the need for GPU acceleration. LightGBM can handle tabular data effectively and capture complex relationships between features and the target variable.

Reasoning-

Accuracy: Gradient boosting algorithms, including LightGBM, are capable of capturing non-linear relationships and interactions between features, which can be valuable for predicting bacterial properties. LightGBM uses a gradient-based optimization algorithm that iteratively minimizes the loss function, resulting in accurate predictions.

Efficiency: LightGBM is designed to be computationally efficient and memory-friendly. It implements techniques like histogram-based gradient boosting and leaf-wise tree growth to minimize memory usage and reduce computational costs. This efficiency allows training and inference to be performed on a potato computer without the need for a GPU.

Handling Tabular Data: LightGBM is well-suited for tabular data, where each column describes a specific property of the bacteria. It can handle a large number of features and automatically handles missing values. It also provides feature importance analysis, which can help identify the most influential features in predicting bacterial properties.

Paper Citation: "LightGBM: A Highly Efficient Gradient Boosting Decision Tree" by Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu (2017)

This paper introduces the LightGBM framework and discusses its efficient implementation of gradient boosting decision trees. It highlights the algorithm's performance advantages, including its accuracy, scalability, and computational efficiency.

In summary, for the task of predicting bacterial properties on a potato computer without GPU access, LightGBM is a suitable choice. It provides a good balance between accuracy and efficiency, making it well-suited for tabular data analysis. The cited paper by Ke et al. provides detailed insights into the LightGBM framework and its benefits.

If you had **access to a GPU cluster**, you could consider using deep learning models for predicting bacterial properties.

I. Specifically, a deep neural network architecture such as a multi-layer perceptron (**MLP**) could be employed.

Reasoning:

Deep Learning Advantages: Deep learning models, such as MLPs and CNNs, have the ability to automatically learn intricate patterns and representations from the data. They can capture complex relationships and extract relevant features, which can be beneficial for predicting bacterial properties from a high-dimensional dataset.

GPU Acceleration: Deep learning models can benefit significantly from GPU acceleration. GPUs excel at parallel processing, which speeds up the training and inference of deep neural networks. With a GPU cluster, you can take advantage of distributed computing and train larger and more complex models, leading to potentially improved accuracy.

Scalability: Deep learning models have the potential to scale well with increasing amounts of data. If your dataset grows in the future, having access to a GPU cluster allows you to efficiently handle larger datasets and potentially improve the model's performance.

Paper Citation: "Deep Residual Learning for Image Recognition" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016)

This influential paper introduced the ResNet architecture, which is a deep residual learning framework that achieved state-of-the-art performance in image recognition tasks. While the paper specifically focuses on image recognition, the principles and advancements discussed are applicable to deep learning models in general.

To summarize, if you had access to a GPU cluster, employing deep learning models such as MLPs could be a viable option for predicting bacterial properties. These models have the ability to learn complex patterns from high-dimensional data and can benefit from GPU acceleration to expedite training and improve accuracy. The cited paper by He et al. provides insights into the advancements in deep learning architectures and their potential benefits.

II. **XGBoost** is another viable option for predicting bacterial properties, even if you have access to a GPU cluster. XGBoost is a powerful gradient boosting algorithm that offers high accuracy and scalability, making it suitable for handling large-scale datasets.

Reasoning:

Accuracy: XGBoost is known for its strong predictive performance. It leverages gradient boosting techniques to iteratively improve the model's accuracy by minimizing the loss function. It handles non-linear relationships between features and the target variable effectively, which can be valuable for predicting bacterial properties accurately.

Scalability: XGBoost is designed to handle large-scale datasets efficiently. It implements parallel processing, allowing it to take advantage of the computational power of GPUs in a cluster environment. With a GPU cluster, XGBoost can process data in a distributed manner, significantly reducing the training time and enabling the use of larger datasets if available.

Feature Importance: XGBoost provides feature importance analysis, which can help identify the most influential features for predicting bacterial properties. This analysis can guide feature selection and engineering efforts, enabling a better understanding of the underlying factors contributing to the prediction task.

Paper Citation: "XGBoost: A Scalable Tree Boosting System" by Tianqi Chen and Carlos Guestrin (2016)

This paper introduces the XGBoost algorithm and discusses its scalability and efficiency. It highlights the benefits of parallelization and tree construction techniques employed by XGBoost to achieve state-of-the-art results in various machine learning tasks.

To summarize, XGBoost is a powerful gradient boosting algorithm that can be a suitable choice for predicting bacterial properties, even with access to a GPU cluster. It offers high accuracy, scalability, and feature importance analysis. The cited paper by Chen and Guestrin provides detailed insights into the XGBoost algorithm and its advantages in terms of scalability and efficiency.

2. PREDICTING NUMBER OF PEOPLE ON THE BEACH

For predicting the number of people on the beach based on weather features, I would suggest using a regression model. Regression models are suitable for predicting numerical values, such as the crowd size in this case. Considering the dataset contains normalized weather features, it's important to choose a model that can effectively capture the relationships between these features and the target variable.

I. **Linear Regression**: Linear regression can be a straightforward and interpretable choice. It assumes a linear relationship between the weather features and the target variable (crowd size). Linear regression estimates the coefficients that represent the impact of each weather feature on the predicted crowd size. This model can provide insights into the importance and directionality of each weather feature's influence on beach attendance.

   Paper: "Least squares optimization with L1-norm regularization" by Tibshirani (1996). This paper introduces the concept of L1-norm regularization in linear regression, which helps improve model accuracy and mitigate overfitting.

However, the features are normalized and the performance of linear regression gets affected due to it. Thus, normalization of features might not have a significant impact on tree-based algorithms like Decision Trees and Random Forests. Tree-based algorithms are not sensitive to the scale or normalization of features because they make splits based on the values and thresholds of individual features.

Normalization is particularly useful for distance-based algorithms or models that rely on optimizing objective functions, such as linear regression or neural networks. These models can be affected by differences in the scale or magnitude of features, and normalization helps to bring all features to a similar scale.

In the case of a dataset with normalized weather features, we can still benefit from tree-based algorithms in the following ways:

- Handling Non-Linear Relationships: Tree-based algorithms are flexible and capable of capturing non-linear relationships between features and the target variable. Even with normalized features, these algorithms can detect complex patterns and interactions in the data.
- Feature Importance Analysis: Tree-based algorithms provide a measure of feature importance, indicating which weather features have the most influence on predicting the crowd size. This analysis can help identify the most significant factors driving beach attendance.
- Robustness to Outliers: Tree-based algorithms can handle outliers in the data, as they do not rely on distance calculations. If your dataset contains outliers or extreme values in the normalized weather features, tree-based algorithms can still provide reliable predictions.
- Ensemble Methods: Random Forest, an ensemble of decision trees, can enhance the performance and robustness of predictions. By combining

multiple decision trees, Random Forest can reduce overfitting and improve generalization.

While normalization may not directly impact the performance of tree-based algorithms, it is still recommended to keep the weather features normalized for consistency and comparability. This ensures that any future modifications or additional features can be incorporated seamlessly into the model without the need for re-normalization.

In summary, even with normalized features, tree-based algorithms can be valuable for modeling the relationship between weather features and beach attendance. They offer flexibility, interpretability, and robustness to outliers, making them viable options for your prediction task.

II. **Decision Trees and Random Forest**: Decision tree-based models, such as Random Forest, can capture nonlinear relationships between the weather features and the target variable. These models can handle interactions and complex patterns in the data. Random Forest, in particular, combines multiple decision trees to improve predictive accuracy and handle feature importance analysis. It can provide valuable insights into the relative importance of different weather features in predicting the crowd size.

Paper: "Random Forests" by Breiman (2001). This influential paper introduces the random forest algorithm, which combines an ensemble of decision trees for improved accuracy and robustness in predictive modeling tasks.

III. **Gradient Boosting** (e.g., XGBoost or LightGBM)

Reasoning: Gradient boosting algorithms can capture complex relationships between weather features and beach attendance. They provide high accuracy and handle non-linear patterns effectively. Training speed depends on the dataset size, but with 20,000 rows, it should be manageable. GPU acceleration is not required.

Papers:

- "XGBoost: A Scalable Tree Boosting System" by Tianqi Chen and Carlos Guestrin (2016)
- "LightGBM: A Highly Efficient Gradient Boosting Decision Tree" by Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu (2017)

Given that the dataset has normalized weather features and the goal is to predict the number of people on the beach, several models can be suitable for this scenario. Here are a few options to consider:

IV. **Support Vector Machines (SVM)**

Reasoning: SVMs aim to find an optimal hyperplane that separates the data based on their features. SVMs with appropriate kernel functions can capture nonlinear relationships between the weather features and beach attendance. They can work well when there is a clear separation between different groups of attendance levels based on weather features.

Paper: "A training algorithm for optimal margin classifiers" by Cortes and Vapnik (1995). This paper presents the Support Vector Machine algorithm and the concept of finding an optimal margin classifier. It discusses the mathematical formulation and the training algorithm for SVMs.

V. **Neural Networks**: Neural networks, such as multi-layer perceptron (MLP) models, can handle non-linear relationships and capture complex patterns in the data. They can be effective when there are intricate interactions between weather features and beach attendance. However, keep in mind that neural networks may require more computational resources and a larger amount of data to train effectively.

Paper: "Deep Learning" by LeCun et al. (2015). This paper provides an overview of deep learning, including the architecture and training algorithms for deep neural networks. It discusses the advantages and applications of deep learning models in various domains, including matrix operations.

Following is the **comparison** of the above models based on accuracy, training time, and other parameters:

- Accuracy:
    - Linear Regression: Linear regression assumes a linear relationship between the input features and the target variable. Its accuracy heavily depends on the linearity assumption and the quality of feature engineering. It may not capture complex non-linear relationships accurately.
    - Decision Trees and Random Forests: Decision trees and random forests can capture non-linear relationships and interactions between features effectively. They can provide good accuracy, especially when there are non-linear dependencies in the data.
    - SVM: SVMs can handle non-linear relationships through the use of kernel functions. By transforming the data into a higher-dimensional space, SVMs can capture complex decision boundaries and provide good accuracy.
    - Neural Networks: Neural networks are known for their ability to capture complex non-linear relationships. With deep architectures and sufficient training, neural networks can achieve high accuracy on a wide range of tasks, including matrix multiplication.
    - Gradient Boosting: Gradient boosting combines weak learners (typically decision trees) to create a strong predictive model. It

iteratively improves the model by focusing on the mistakes of previous iterations. Gradient boosting can achieve high accuracy, often outperforming individual decision trees.

- Training Time:
  - Linear Regression: Linear regression has a fast training time since it involves solving a closed-form equation or performing simple iterative updates.
  - Decision Trees and Random Forests: Training time for decision trees and random forests can be higher compared to linear regression, especially when the dataset is large and the trees are deep. Random forests require training multiple trees, which increases training time.
  - SVM: SVM training time can be relatively high, especially for large datasets. The training time scales quadratically with the number of samples, making it slower for large datasets.
  - Neural Networks: Training time for neural networks can be time-consuming, especially for deep architectures and large datasets. The training time depends on factors such as the network size, the number of layers, and the complexity of the problem.
  - Gradient Boosting: Gradient boosting can have a higher training time compared to individual decision trees since it builds the model iteratively. Each iteration requires training a weak learner and updating the model.

- Interpretability:
  - Linear Regression: Linear regression provides interpretability as the coefficients of the linear equation indicate the influence of each feature on the target variable.
  - Decision Trees and Random Forests: Decision trees can provide interpretable results as the splits and leaf nodes represent logical rules. Random forests, being an ensemble of decision trees, are less interpretable but can still provide insights into feature importance.
  - SVM: SVMs are less interpretable compared to linear regression or decision trees, as they rely on complex mathematical transformations in higher-dimensional spaces.
  - Neural Networks: Neural networks are generally considered less interpretable, especially with deep architectures. The network weights and connections are not easily interpretable in terms of feature importance.
  - Gradient Boosting: Similar to decision trees, gradient boosting can provide insights into feature importance, but as an ensemble method, it may be less interpretable than individual decision trees.

- Handling Non-linearity:
    - Linear Regression: Linear regression assumes a linear relationship between variables and may not handle non-linearity well without feature engineering or transformations.
    - Decision Trees and Random Forests: Decision trees and random forests can handle non-linear relationships between variables effectively without explicit feature engineering.
    - SVM: SVMs can handle non-linear relationships by using kernel functions to transform the data into higher-dimensional spaces.
    - Neural Networks: Neural networks are highly effective in capturing non-linear relationships due to their ability to model complex interactions and hierarchies between features.
    - Gradient Boosting: Gradient boosting can capture non-linear relationships, as it combines multiple weak learners to form

Consider the interpretability of the model, the dataset size, the complexity of the relationship between weather features and beach attendance, and the available computational resources when selecting the most appropriate model. It's recommended to experiment with different models, evaluate their performance using appropriate metrics, and choose the one that provides the best trade-off between accuracy and interpretability for your specific scenario.

3. TEXT - IMAGE SEARCH ENGINE

I. **ViLBERT**: It is a multimodal framework that integrates both textual and visual information. To train representations that capture the semantic and visual information of the input data, it combines masked language modeling with image-text matching goals. The ViLBERT model is particularly suited for the task of text-image search, where the objective is to link verbal descriptions with associated visual content. It merges the two modalities into a single framework that enables precise matching and the retrieval of pertinent data. However, it's crucial to keep in mind that putting multimodal models into use and training them can be computationally taxing and expensive.

Other multimodal models, such VisualBERT and UNITER, are also available and have demonstrated good outcomes in vision-and-language challenges. The model selected will rely on particular needs, resources available, and desired trade-offs between accuracy and training speed.

Paper: "ViLBERT: Pre Training Task-Agnostic Visio Linguistic Representations for Vision-and-Language Tasks" by Jiasen Lu et al. (2019)

II. **Combining Natural Language Processing (NLP) methods with Convolutional Neural Networks (CNN)** is another method for developing a text-image search engine. In this method, the image and text inputs are processed independently, and the search and retrieval tasks are subsequently completed by combining the extracted characteristics.

Image Processing with **CNN**: A CNN is used to process the image input and extract visual information. A pre-trained model such as VGGNet, ResNet, or EfficientNet could be used as the CNN; these models have demonstrated good performance on picture classification and feature extraction tasks.

Text Processing with **NLP**: NLP methods are used to preprocess text input such as captions or textual descriptions. Tokenization, stemming/lemmatization, and encoding the text into a numerical form (such as word embeddings or TF-IDF vectors) are all included in this.

Feature Fusion: A joint representation is created by fusing together the encoded text representations from NLP with the extracted visual characteristics from CNN. The feature vectors can be combined, fed into a fusion layer, or attention techniques can be used to dynamically balance the weight of each modality.

Search and Retrieval: The search engine compares the representation of the query to the representations of images or text in the dataset given an input query (either image or text). Cosine similarity or Euclidean distance are two methods used to calculate how similar the representations are to one another. Search results show the most pertinent matches.

Paper for CNN: "Very Deep Convolutional Networks for Large-Scale Image Recognition" by Simonyan and Zisserman (2014) or "Deep Residual Learning for Image Recognition" by He et al. (2016)
Paper NLP: "Distributed Representations of Words and Phrases and their Compositionality" by Mikolov et al. (2013) or "GloVe: Global Vectors for Word Representation" by Pennington et al. (2014)

III. **Visual-Semantic Embedding (VSE) model**. The VSE model aims to learn a joint embedding space where images and textual descriptions are represented as continuous vectors, such that similar images and descriptions are closer together in this embedding space.

Methodology:
The general methodology for building a text-image search engine using the VSE model involves the following steps:

Data preprocessing: Preprocess the MS-COCO 2014 dataset to extract image features and textual descriptions. For images, you can use pre-trained convolutional neural networks (CNNs) like ResNet or Inception to extract visual features. For textual descriptions, techniques like tokenization, stemming, and removing stop words can be applied.

Model training:

- Image Encoder: Train a CNN-based image encoder that takes an image as input and outputs a fixed-length visual feature vector. This encoder learns to capture the visual content and semantics of the image.
- Text Encoder: Train a recurrent neural network (RNN)-based text encoder that takes a textual description as input and generates a fixed-length text feature vector. This encoder learns to capture the semantic information in the text.
- Cross-Modal Matching: Use a contrastive loss function to train the model to minimize the distance between matching image-text pairs and maximize the distance between non-matching pairs. This encourages the model to learn a shared embedding space where similar images and descriptions are close together.
- Image-to-Text: Given an input image, pass it through the image encoder to obtain its visual feature vector. Then, compare this vector with the text feature vectors of all textual descriptions using similarity measures like cosine similarity or Euclidean distance. Retrieve the top-k most relevant textual descriptions based on the similarity score.
- Text-to-Image: Given an input textual description, pass it through the text encoder to obtain its text feature vector. Compare this vector with the visual feature vectors of all images using similarity measures. Retrieve the top-k most relevant images based on the similarity score.

Loss function:
The most commonly used loss function for training the VSE model is the contrastive loss function, which encourages similar image-text pairs to be close together and dissimilar pairs to be far apart in the embedding space. The contrastive loss can be formulated as a margin-based ranking loss, where the model aims to minimize the distance between matching pairs and maximize the distance between non-matching pairs.

Paper:

F. Xu et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In Proceedings of the International Conference on Machine Learning (ICML), 2015. This above paper provides insights into a related approach that incorporates visual attention mechanisms for generating image captions. Although it is not specifically focused on text-image search, it discusses relevant techniques and can serve as a valuable reference.

Overall, building a text-image search engine using the MS-COCO 2014 dataset involves training a multimodal model, such as the Visual-Semantic Embedding (VSE) model, to learn a shared embedding space for images and textual descriptions. The model is trained using a contrastive loss function and can be used for both image-to-text and text-to-image retrieval by comparing feature vectors in the embedding space.

Here is a comparison between the above models:

- Performance:

    1. ViLBERT: ViLBERT has achieved state-of-the-art performance on various vision-and-language tasks, including the MS-COCO dataset. It can generate relevant textual descriptions given an image and vice versa, making it suitable for a text-image search engine. The paper "ViLBERT: Pre Training Task-Agnostic Visio Linguistic Representations for Vision-and-Language Tasks" by Lu et al. (2019) showcases its performance.
    2. VSE: VSE has demonstrated strong performance in cross-modal retrieval tasks, including text-image searches. It can provide accurate and relevant textual descriptions for images and vice versa. The paper "From Captions to Visual Concepts and Back" by Kiros et al. (2014) presents its capabilities.
    3. Combination of CNN and NLP: The performance of this approach depends on the specific implementation and architecture. By combining CNN and NLP techniques, it is possible to achieve reasonable performance in text-image retrieval tasks. However, it may not match the state-of-the-art performance achieved by dedicated multimodal models like ViLBERT or VSE.

- Model Complexity:

    1. ViLBERT: ViLBERT is a complex model that combines Visual BERT and Language BERT. It requires substantial computational resources for training and inference due to its joint processing of visual and textual inputs. The complexity of the model allows for more nuanced interactions between modalities.
    2. VSE: VSE is a simpler approach compared to ViLBERT. It uses separate models for image and text embeddings and aligns them in a shared embedding space. This simplicity makes it faster to train and less computationally demanding than ViLBERT.
    3. Combination of CNN and NLP: The model complexity of this approach depends on the specific CNN and NLP architectures used. It can range from relatively simple combinations to more intricate architectures. Generally, it is less complex than ViLBERT but may still require significant resources depending on the chosen components.

- Dataset Availability:

    1. ViLBERT: The ViLBERT paper demonstrates its performance using the MS-COCO dataset, which is widely used for vision and language tasks. The dataset is publicly available, and preprocessed versions for specific tasks are accessible.
    2. VSE: The VSE paper also utilizes the MS-COCO dataset, which is a popular benchmark dataset for cross-modal retrieval tasks. The dataset is publicly available and widely used in the research community.
    3. Combination of CNN and NLP: The MS-COCO dataset is also applicable for training a combination of CNN and NLP models. As a widely used dataset, it is accessible for training and evaluation purposes.

- Development Timeline:
    1. ViLBERT: The development timeline for ViLBERT can be relatively longer due to its complexity and resource requirements. It may involve substantial time for model design, training, and fine-tuning to achieve optimal performance.
    2. VSE: The development timeline for VSE can be shorter compared to ViLBERT. Its simpler architecture and training process enable faster iterations and experimentation.
    3. Combination of CNN and NLP: The development timeline for a combination of CNN and NLP models can vary based on the chosen components and the desired level of performance. It can be shorter than ViLBERT but longer than VSE, depending on the complexity of the chosen CNN and NLP architectures.

Considering the parameters mentioned, if high performance is the primary consideration, ViLBERT would be the recommended choice, followed by VSE.

4. MATRIX MULTIPLICATION

One method for accelerating matrix multiplication would be to utilize a library like **cuBLAS** with GPU acceleration. Fast matrix operations, including matrix multiplication, are provided via the highly optimized GPU-accelerated library cuBLAS, which is made available by NVIDIA. It accelerates computations significantly faster than conventional CPU-based methods by taking advantage of GPUs' parallel processing capabilities.

Paper: "Optimizing matrix multiplication for CUDA-enabled GPU" by Li et al. (2011)

When performing matrix multiplication operations, cuBLAS can significantly speed things up. By **utilizing the parallelism built into GPU architectures**, the library offers optimized implementations of matrix operations and is built to make optimum use of GPU resources. Compared to CPU-based computations, this leads to quicker training times.

Since matrix multiplication is a clearly defined mathematical activity, the particular model or method employed may not make a major difference when optimizing for accuracy in matrix multiplication.
Paper: "Deep Residual Learning for Image Recognition" by He et al. (2016)

Using GPU acceleration with cuBLAS substantially reduces the training period for matrix multiplication. The speedup obtained depends on various elements, including the size of the matrices, the difficulty of the computations required, and the GPU's processing capacity. The ability of GPUs to execute matrix operations in parallel typically results in significant speed advantages over CPU-based methods.

It is more difficult to optimize for speed when GPU resources are not available. Alternative strategies, including multi-core CPUs and parallelization techniques, can be used in such circumstances. Performance on CPU-based systems can be enhanced by parallelizing matrix multiplication utilizing multi-threading, **SIMD** (Single Instruction, Multiple Data), or distributed computing techniques. It is crucial to remember that for matrix multiplication, CPU-based parallelization might not be as quick as GPU acceleration.

Paper: "Parallelizing Matrix Operations on Shared Memory Systems" by Luszczek et al. (2005)

To summarize, when optimizing for speed in matrix multiplication:

- Using the cuBLAS library with GPU acceleration results in considerable speed gains.
- Given that matrix multiplication is a well defined operation, the choice of model or method is less important.
- The accuracy of matrix multiplication depends on the algorithm's correctness and the accuracy of the numbers.
- GPU acceleration with cuBLAS dramatically increases training speed.

    - Parallelization strategies on multi-core CPUs can be taken into consideration if GPU resources are not available, however they might not be as fast as GPU acceleration.

Question Answer:

- For optimizing speed in matrix multiplication with a dataset consisting of two random 3x3 matrices and their matrix product, the most suitable technique would be optimized linear algebra libraries like MKL or cuBLAS. These libraries are specifically designed to efficiently perform matrix operations and can leverage hardware acceleration, such as GPUs, if available. They provide high computational efficiency and are well-suited for speeding up matrix multiplication on both CPU and GPU platforms.

- If the goal is to optimize for accuracy, deep learning-based approaches such as multi-layer perceptron (MLP) or convolutional neural networks (CNN) can be employed. These models can learn complex patterns and relationships in the data, providing accurate matrix multiplication. However, it's important to note that the dataset size (3x3 matrices) may not be ideal for deep learning approaches, as they typically require larger datasets to generalize well.

- WHY not applicable
  Regarding the choice of Strassen's algorithm and Winograd's algorithm, these algorithms are more effective for larger matrices, typically beyond 3x3. Since the given dataset consists of small matrices, their benefit might be limited. The overhead associated with these algorithms, such as recursion and preprocessing, might outweigh the computational savings for such small matrices.

In summary, for optimizing speed, MKL or cuBLAS would be the best options, while for optimizing accuracy, deep learning-based approaches can be explored if the dataset size is sufficient.