Discover Computing

Research

Leveraging transformers architectures and augmentation for efficient classification of fasteners and natural language searches

Nino Cauli[1] · Marco Murgia[1] · Diego Reforgiato Recupero[1,2] · Giuseppe Scarpi[2]

Received: 12 September 2023 / Accepted: 17 May 2024
Published online: 27 May 2024

Abstract
A primary concern in the realm of mechanical engineering is to ensure the efficient and effective data entry of hardware devices. Fasteners are mechanical tools that rigidly connect or affix two surfaces or objects together. They are small and often different fasteners might look similar; it is therefore a long and prone-to-risk procedure to manually analyze them to classify and store their related information. With the widespread diffusion of AI frameworks in several domains, equipment manufacturers started to rely on AI technologies for these heavy tasks. Automatically classifying fasteners by type and extracting metadata from natural language questions are important tasks that fastener manufacturers and suppliers encounter. In this paper, we address these challenges. To address the first task, we introduce an augmentation methodology that starts with a small set of 3D models representing each of the 21 types of fasteners we aim to classify. This methodology efficiently generates multiple 2D images from these models. Next, we train a vision transformer using the collected data to address a single-label multi-class classification task. For the second task, we introduce a prompt-engineering technique designed for conversational agents. This technique leverages in-context knowledge to extract (metadata field, value) pairs from natural language questions. Subsequently, we tackle a question-answering task to the description fields of the extracted fasteners. Our evaluation demonstrates the effectiveness of both approaches, surpassing the baselines we tested.

Keywords Recommendation · Transformers · Natural language processing · E-Recruitment

1 Introduction

A primary concern in the realm of mechanical engineering is to ensure the efficient and effective data entry of hardware devices. Fasteners such as bolts, nuts, and washers are devices that attach one thing to another or hold something in place. The cost of a single fastener is typically just a few cents or fractions of a Euro. Due to their small size and often similar appearance, manually classifying these elements and storing their related information is a laborious and error-prone procedure.

Historically, the *automatic optical inspection* [1] has been the preferred solution for the analysis of fasteners. This involves specialized machinery capturing images of the components under examination, comparing them to reference

Nino Cauli, Marco Murgia, Diego Reforgiato Recupero and Giuseppe Scarpi have contributed equally to this work.

✉ Diego Reforgiato Recupero, diego.reforgiato@unica.it; Nino Cauli, nino.cauli@unica.it; Marco Murgia, m.murgia98@studenti.unica.it; Giuseppe Scarpi, giuseppe.scarpi@r2msolution.com | [1]Department of Mathematics and Computer Science, University of Cagliari, Via Ospedale 72, Cagliari 09124, Italy. [2]R2M Solution S.r.l., Via Fratelli Cuzio, 42, Pavia 27100, Italy.

models, and providing an automatic classification of their types using unsupervised approaches for subsequent analysis of various kinds.

However, with the increasing standards for quality, equipment manufacturers are increasingly shifting towards AI-based systems. These systems offer improved performance related to the capability to identify the fastener type and even respond to questions about them.

A recurring challenge in working with AI systems is the requirement for large datasets containing diverse images of each fastener type for effective classification, as noted by manufacturers [2]. Managing diverse fastener metadata (e.g., type, description, etc.) typically involves utilizing a database. However, this alone is insufficient. Adding new samples to the database can be labor-intensive and error-prone. Additionally, retrieving existing samples is often time-consuming, necessitating the specification of numerous details.

Building upon these open issues, we endeavored to develop original solutions by leveraging cutting-edge technologies to support operators. It is important to emphasize that human involvement remains irreplaceable in these tasks. However, AI can play a vital role in assisting human operators, streamlining processes, and reducing the likelihood of errors.

In line with this approach, this paper presents a novel methodology we have developed to automate the classification of fasteners and efficiently retrieve candidate fasteners based on natural language queries. To the best of our knowledge, we are the first to propose a solution addressing these two tasks specifically within the domain of fasteners.

During the data entry phase, for each fastener, we equipped operators with a *Vision Transformer* (ViT) [3][1] specifically trained to identify the type of fastener. The type was just one of several metadata attributes included for each fastener in the database. As we will discuss later, training the ViT posed challenges due to the limited availability of non-commercial fastener images online. We adopted a novel approach that proved highly effective and has potential applicability in similar contexts once validated. Specifically, we generated a large dataset of images corresponding to 21 different types of fasteners. To accomplish this, we began with a small number of 3D image samples found online for each fastener type. Using the Unity framework, we created multiple 2D versions by systematically varying characteristics such as background, scene illumination, resolution, and more.

Next, leveraging a database containing various fasteners with associated metadata (including type and description), we addressed a question-answering task when presented with a natural language query about a fastener. Initially, we retrieved a list of appropriate candidate fasteners based on the query, followed by applying state-of-the-art transformers fine-tuned for question-answering to extract relevant information. This list comprised fasteners whose metadata aligned with details extracted from the natural language question. For example, from the question *"Can you describe where M10 copper bolts are used?"*, the system should retrieve the following {metadata field, value} pairs: *{fastener, bolt}, {materials, copper}, {thread_type, M10}*.

To streamline data retrieval, we employ effective prompt engineering techniques [4] tailored for use within conversational agents, utilizing domain-specific context. The baseline against which we compared our proposed technique involves directly generating SQL queries from a natural language sentence using fine-tuned transformers [5].

Our evaluation of the image classification task and metadata retrieval demonstrates two key findings: (i) the collected dataset and employed vision transformer achieve high F1 values; (ii) the effectiveness of our proposed approach for extracting metadata from natural language queries, as evidenced by results from the question-answering task applied to fastener descriptions. As additional contributions, we are releasing the set of 2D images produced, along with the questions used for question-answering and the prompts employed.

The remainder of this paper is organized as follows. Section 2 provides past works about two concepts: (i) the application of vision transformers to image classification tasks and (ii) the different ways to employ transformers (through fine-tuning for machine translation tasks) and large language models (through appropriate prompt engineering strategies) for translating natural language questions in SQL queries. Section 3 discusses the two research tasks we target in this paper. Regarding the first task, Sects. 4 and 5 detail the innovative methodology used to collect the fastener dataset and the vision transformer along with its parameters that were utilized. Sect. 6 outlines the approach we proposed to address the second task. In Sect. 7 we report the evaluation we have carried out for the two tasks. Finally, in Sect. 8, we conclude the paper with final remarks and discuss future directions.

[1] https://huggingface.co/docs/transformers/model_doc/vit.

2 Related work

In this section, we will illustrate recent works about vision transformers used for image classification, fine-tuned transformers for translating natural language questions in SQL, and prompt engineering techniques to extract metadata from text using large language models [6].

The seminal work about vision transformers has been performed by authors in [3]. They applied a typical Transformer directly to images. To do this, they divided an image into patches and fed the series of their linear embeddings into a Transformer. Image patches are handled in the same manner as tokens (words) in Natural Language Processing applications. Moreover, the authors used supervised learning to train the model for image classification and proved that the tested vision transformer outperformed competitors on different datasets such as ImageNet, ImageNet-ReaL, CIFAR, and VTAB. Other authors have then applied vision transformers for video classification [7]. From the input video, their model extracts spatio-temporal tokens, which are then encoded by a number of transformer layers. Different variations of the proposed model are tested to factorize the spatial and temporal dimensions of the input with the aim to manage the lengthy token sequences found in the video. The work published in [8] describes a review of papers about the vision transformers distinguishing each of them by the addressed task. The advantages and disadvantages of each mentioned method and future research directions for vision transformers are analyzed and discussed. Other relevant works about vision transformers are [9–11].

Due to its practical applicability in creating natural language interfaces to database systems and its capacity to translate the semantics of natural language into SQL queries, text-to-SQL [12] has gained interest from both the natural language processing and database sectors. Encoding natural language meaning, decoding to SQL queries, and translating the semantics between these two forms are the main issues in text-to-SQL. In general, given an input sentence in natural language, two techniques that are commonly used to extract information from a database are: (i) fine-tuning a transformer for a dedicated machine translation task (from natural language text to SQL); and (ii) using in-context learning and asking the conversational agent to return the (database field, value) pairs out of a sentence in natural language. Authors in [13] presented a systematic overview examining current developments in text-to-SQL for datasets, methodologies, and assessment. Whether or not one strategy is better than the other is still a point where different researchers are working. In [4], authors tried to shed light on this problem and compared the generalization of few-shot fine-tuning and in-context learning to challenge datasets. The outcome of the analysis demonstrates that well-tuned language models can in fact generalize effectively across domains. Moreover, both methods generalize similarly; they display considerable variation and depend on factors like model size and the number of examples, highlighting the fact that effective task adaptation is still a difficult process. Other authors heavily analyzed in-context learning and found different use cases to jailbreak questions given to ChatGPT across eight prohibited scenarios [14]. The study emphasizes the significance of prompt engineering in jailbreaking large language models and explores the difficulties in creating and preventing effective jailbreak prompts. Other authors have described in [15] a catalog of prompt engineering techniques to be used to solve common problems during an interaction with large language models. First, they offer a framework for identifying patterns for organizing prompts to address a variety of problems so that they can be customized for use in various domains. Second, they offer a list of patterns that have been used to enhance the results of large language model talks. Thirdly, they describe how many patterns may be combined to create prompts and provide examples of how some patterns can be improved by being combined with others.

To the best of our knowledge and distinct from existing literature, we are the first to apply vision transformer technology to the single-label multi-class classification of fasteners. We trained this model using a gold standard dataset of 2D images representing 21 different fastener types, generated from a few samples of 3D models for each type. The high F1 scores achieved in our evaluation against real-world 2D images underscore the effectiveness of our augmentation methodology.

Subsequently, we utilized various prompt engineering techniques to efficiently extract (metadata field, value) pairs from natural language questions related to a collection of fasteners stored in a database alongside their metadata.

3 Tasks definition

This paper sets out to address two distinct tasks. The initial task pertains to a single-label multiclass classification problem within the domain of object categorization. More in detail, our objective is to categorize 21 specific fasteners[2]:

[2] A *fastener* is a mechanical device, often made of metal or plastic, that is designed to physically connect or secure two or more items. Common examples include bolts and nuts.

- Bolts with cylindrical head and three types of insets: slot (DIN[3]: 84), shoulder-slot (DIN: 923), hex (DIN: 912);
- Six other kinds of bolts: countersunk head with hex inset (DIN: 7991), eyed (DIN: 444), hexagonal head (DIN: 914), raised countersunk with slot inset (DIN: 964), set screw bolt (DIN: 417), thread insert bolt (DIN: 7965);
- Five types of nuts: castellated (DIN: 935), grooved (slotted) (DIN: 1804), hexagonal self-locking (DIN: 985), winged (DIN: 315), hexagonal (DIN: 936);
- Blind rivet (DIN: 7337);
- Four securing elements: circling for bore (DIN: 472), circling for shaft (DIN: 471), serrated internal (DIN: 6798), with tab lock (DIN: 462);
- Split-pin (DIN: 94);
- Plain washer (DIN: 6916).

Therefore, each object in the dataset can be assigned to only one of the 21 categories. The objects we are trying to classify are typically small and may vary in minor details.

The second task we are addressing involves retrieving (metadata field, value) pairs associated with fasteners stored in a database. This is achieved by providing a natural language query that may contain various expressions and language nuances indicating one or more characteristics.

We are not focusing on the database and its schema because the methodology we propose is domain-agnostic and schema-agnostic.

In the rest of the paper, we will refer to the two tasks as task 1 and task 2.

4 The collected dataset of images

As previously mentioned, the first task falls within the scope of object categorization. It aims to differentiate among 21 types of fasteners, which often differ by small details.

The initial challenge we encountered was related to the data collection for training the transformer. This was primarily due to the fact that a substantial portion of high-quality fastener images are copyrighted.

Data augmentation has been widely applied in several domains in the literature [16–18]. Therefore we decided to generate our own synthetic dataset from open-source 3D models using open-source applications. However, we recognized that this approach would introduce an extra variable as training a ViT with synthetic images to categorize real ones can be tricky.

Although the methodology is general and anyone can use any off-the-shelf tool, in our specific case we used three software tools: Unity3D, a versatile graphics engine popular for video game development but adaptable to many other applications; Blender, a well-known open-source software for 3D modeling; and FreeCAD, an open-source 3D CAD. Of course, any other software can be adopted, provided it can execute the operations that we describe in the rest of the paragraph.

As a first step, we selected publicly available CAD models[4] (files with.igs and.stl extensions) for each fastener of interest. These had to be converted to.fbx files for compatibility with Unity3D. This involved an intermediary conversion of the CAD files into.obj files using FreeCAD, followed by a transformation of these into.fbx files using Blender. The conversion to.obj was unnecessary for.stl files since Blender natively supports the.stl format.

Unity offers several advantages over other 3D rendering tools:

- It possesses a powerful and user-friendly interface designed for a wide range of users, including designers, programmers, artists, and researchers.
- In addition to the graphical user interface, it provides a powerful C# scripting API that allows for quick program organization. It enables easy modification of objects in the scene through scripting, facilitating procedural loops to randomize generated images. This includes adjusting camera parameters and position, light position and intensity, objects' pose, size and material, background, and more.

[3] https://en.wikipedia.org/wiki/List_of_DIN_standards.

[4] All these models are made available under the Creative Commons Attribution license and were downloaded from http://grabcad.com. We extend our acknowledgments to the authors John Fall, Eira, Binilbabu, Rusty Smith, and Anderson Alfonso Silva.

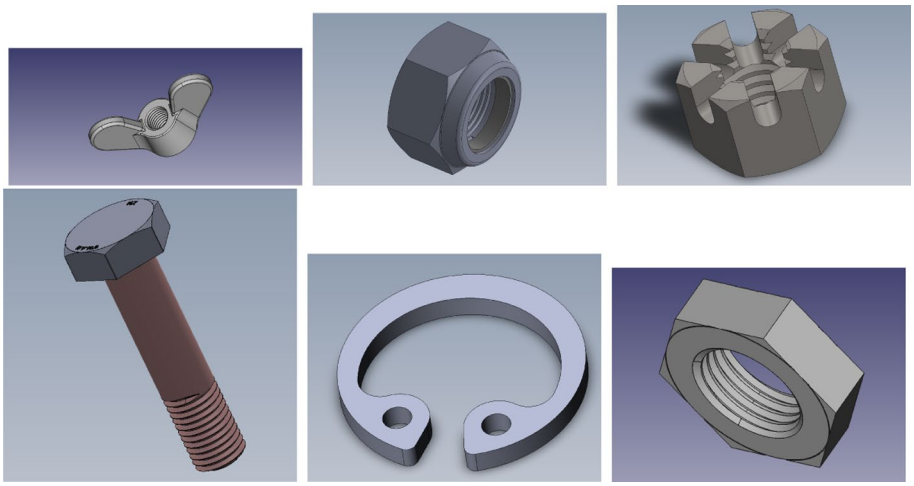

Fig. 1 Example of six 3D CAD models (winged nut, self-locking nut, castellated nut, hex bolt, seeger washer, hex nut)

- Unity offers various render pipelines, such as Built-in, Universal Render Pipeline, and High Definition Render Pipeline, which provide high flexibility in rendering quality. They also allow for the customization of shaders in use. Users can choose the appropriate trade-off between rendering speed and quality to obtain photorealistic images.
- Unity is free to use and provides a vast database of free assets in its asset store.
- Unity is widely utilized in computer vision research, and numerous plugins exist to assist researchers, such as the Perception Package, Synthetic Humans, and MLAgents.

Next, we created a script for Unity3D to process 3D fastener models and generate a variety of unique 2D images from each 3D representation.

More in detail, the features we were able to vary for each 3D image were:

- The angle from which to take an image, either fixed or random;
- The distance of the camera from the object;
- The background against which the photo is taken;
- The intensity, direction, and color of the illumination of the scene;
- The resolution of the generated 2D synthetic images.

We generated around 4000 images for each of the 21 types of fasteners thus collecting 84000 samples.[5]

In Fig. 1 we illustrate six different cad models for six fasteners whereas Fig. 2 depicts one of the 4000 generated images for each of them. Figure 3, in particular, shows 6 different generated 2D images for the first fastener shown in Fig. 2. From Fig. 3, it is evident that there is randomization in the camera position and distance (resulting in changes in the position and size of the fastener in each image), as well as in the illumination direction, intensity, and color (leading to variations in the size and color of reflections in each image).

An additional post-processing step was applied to the generated 2D image dataset to correct instances where the fastener was not centered in the image.

5 The used vision transformer

As far as task 1 is concerned, we used the vision transformer "vit-base-patch16–384", pre-trained on the ImageNet-21k dataset, which includes 14 million images and 21843 classes at a resolution of 224×224.[6]

[5] The set of images that we have generated is freely available at https://drive.google.com/file/d/1eOjZwHwMEjO8SRMDKMi5hH7nW-o88AVB/view?usp=sharing.

[6] https://www.image-net.org/.

Fig. 2 Generated 2D images from the CAD models of Fig. 2

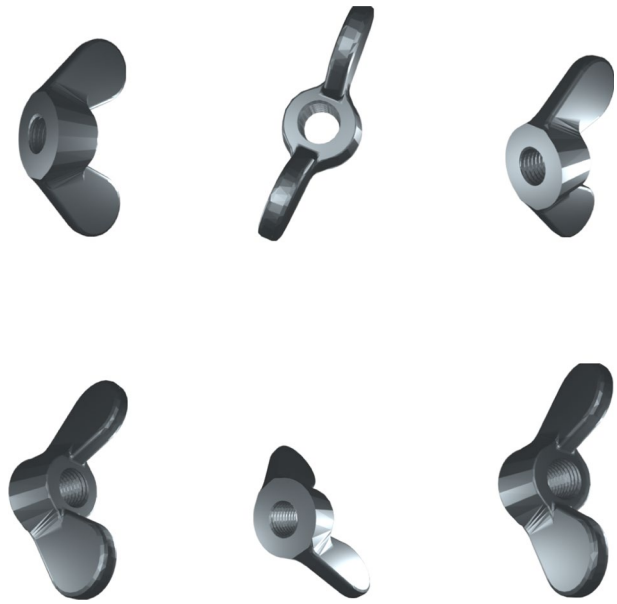

Fig. 3 Generated 2D images from the winged nut model of Fig. 1

The architecture of the used vision transformer is shown in Fig. 4. Each image is split into patches (in this case 16 × 16 resolution). Linear embeddings are created from these patches, and positional embeddings are then added to retain positional information. Standard learnable 1D position embeddings have been employed. Next, the classical transformer encoder processes the resulting vector sequence as input. A token ([cls]) is then appended to the sequence to prepare it for a classification task.

Note that the employed model ("vit-base-patch16–384k") does not provide any fine-tuned heads. However, the model includes the pre-trained pooler, which can be used for downstream tasks such as image classification. Therefore, the pre-training of the model is useful for learning an inner representation of images that can then be used to extract features useful for downstream tasks.

More details about this transformer are publicly available at [19].[7]

As far as fine-tuning is concerned, we chose $2e^{-5}$ as the learning rate, weight decay equal to 0.1, batch size equal to 10 and 5 epochs. The 84000 image samples we collected and mentioned in Sect. 4 have been fed to the transformer for the fine-tuning step. We reserved 20% of the training set for the validation set.

[7] https://huggingface.co/google/vit-base-patch16-384.

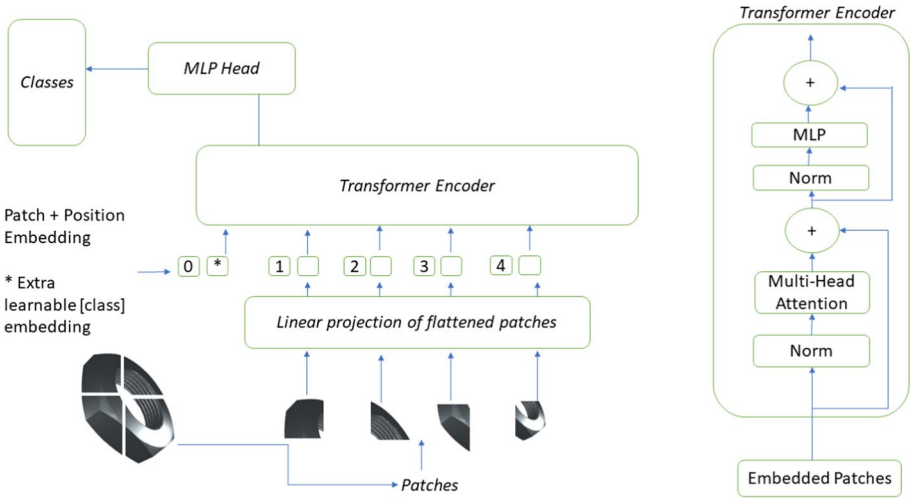

Fig. 4 Architecture of the used ViT (taken from [19])

6 Extracting textual information

For task 2, our goal was to extract (metadata field, value) pairs from natural language queries related to fasteners that are assumed to be stored in the database. In this particular context, we assume that the database was populated with fasteners and their metadata. However, it should be noted that our methodology is universal, not contingent upon a specific database schema. In the remainder of the paper, we will proceed under the assumption that we have stored in a database a set of fasteners along with their metadata including binary, non-binary fields, and one description. The binary fields are restricted to true/false values, while non-binary fields are related to a finite list of values. The description is a textual field that we will later use for question-answering. Our search methodology leverages the *OpenAI GPT-3.5 turbo engine*, which is tailored through meticulous prompt engineering to deliver the most efficacious outcomes. This part posed a considerable challenge, given that users could stipulate any level of complexity in their query, constraining the desired values of one or multiple fields.

The pipeline we have created assumes that:

- The search is restricted to non-text columns (binary and non-binary fields);
- For each non-text column, we describe its allowed values with the format {Column Name, Value 1 | Value 2 | ...}
- The database already contains items describing various fasteners, each of them with the relevant fields correctly populated as previously mentioned.

With these assumptions, our pipeline works as follows:

1. With appropriate prompt engineering we instruct the conversational agent (in our case we used ChatGPT) about the existence of the columns of our database and all the values that each column may contain. This task can be done only once unless the metadata changes;
2. The user asks a question in natural language;
3. We then ask the conversational agent to retrieve all the key-value pairs {Column Name, values} from the input user question. In other words, we are using the conversational agent as a very smart query analyzer and preprocessor.
4. We extract from the database the items that satisfy all the pairs {Column Name, values};

6.1 In-context learning

In this paragraph, we will give technical details about each step of the proposed pipeline.

In step (1), we explain ChatGPT the task to perform, namely to extract key-value pairs of our interest from questions in natural language.

Here are the prompts that we used (only the relevant parts are shown)[8]:

Human I will propose you a list of metadata with related acceptable values. The metadata describe some features of metallic fasteners. The format of the metadata is: Metadatum: <name of metadatum> Values: <comma delimited list of values> Example of question: <an example of question that a human can do in natural language> Correct answer: <list of metadata contained in the question with their respective value(s)> The difficulty is that a human will not put the metadata names in the query. For example, if thread_code defines the type of thread in a bolt, a human would never say "find bolts with thread_code = M3". They will instead ask "find all M3 bolts".

GPT Sure, I can help you with that. Please provide me with the metadata and their acceptable values, along with an example question, and I'll assist you in constructing the correct answer.

Human: Metadatum: fasteners Values: bolt, nut, rivet, screw, securing element, threaded plug, threaded rod, washer Example of question: find all bolts Correct answer: {fastener:bolt}

Metadatum: material Values: steel, iron, copper, brass Example of question: retrieve fasteners in brass Correct answer: {material:brass}

Metadatum: turned_stamped Values: turned, stamped Example of question: get all the steel nuts stamped Correct answer: {fastener:nut material:steel turned_stamped:stamped}

Metadatum: is_coated Values: coated, non coated Example of question: get all screws in copper non coated Correct answer: {fastener:screw material:copper is_coated:non coated}

Specialized metadata when metadatum fasteners = bolt

Metadatum: thread_type Values: M<integer from 1 to 32>, ISO<integer from 1 to 32>, ANSI <any possible ANSI code for threads> Example of question: search all the bolts with thread 0–80 2B Correct answer: {fastener:bolt thread_type:ANSI 0–80 2B}

Metadatum: head_type Values: eye, cylinder, button, countersunk, round, ring, pan, winged, T-head, raised countersunk, knurled, none Example of question: find M4 bolts with countersunk head Correct answer: {fastener:bolt thread_type:M4, head_type:countersunk}

Metadatum: slot type Values: Straight, Phillips, Cross, Square, Torx, Hex, Pozi-drive, Robertson, Tri-wings, Spanner, Clutch, JIS, Pentalobe Example of question: find M3 Torx bolts in copper with cylindrical head Correct answer: {fastener:bolt thread_type:M3, head_type:cylindrical, slot_type:Torx}

Specialized metadata when "fasteners" = nut

Metadatum: thread_type Values: M<integer from 1 to 32>, ISO<integer from 1 to 32>, ANSI <any possible ANSI code for threads> Example of question: find M6 nuts Correct answer: {fastener:nut thread_type:M6}

Metadatum: nut_type Values: hexagon, flanged, capped, winged, caged, knurled, castellated, squared, grooved, ringed, with washer, drive-in Example of question: find flanged M8 nuts Correct answer: {fastener:nut thread_type:M8 nut_type:flanged}

Specialized metadata when "fasteners" = washer Metadatum: washer_type Values: Standard, rosette (also called countersunk), sealing, squared, conical, contact disk Example of question: check if the database contains squared or conical copper washers Correct answer: {fastener:washer material:copper washer_type:squared,conical}

Metadatum: diameter_ext Values: <real from 1 to 50> Example: find steel washers with external diameter 22 Correct answer: {fastener:washer material:steel diameter_ext:22}

Metadatum: diameter_int Values: <real from 1 to 50> Example: find iron washers with external diameter 22 and internal 10, and stamped Correct answer: {fastener:washer material:iron turned_stamped stamped, diameter_ext:22, diameter_int:10}

Metadatum: is_knurled Values: knurled, not knurled Example: extract from DB all knurled washer in steel Correct answer: {fastener:washer material:steel is_knurled:knurled}

GPT: (the answer confirms that GPT understood the task).

Human: I will pass you some new questions. Can you now extract the metadata and their values? I would get the answers in format {metadatum: comma-delimited values}, for example: {fasteners: nut, washer}

[8] The complete prompts are publicly accessible at http://192.167.149.18/promptsFastenersGPT.txt.

For steps 2) and 3), the user proposes questions in natural language and ChatGTP extracts the corresponding pairs. For example:

Human: Check if the database contains squared or conical copper washers.

GPT: {fasteners: washer}, {material: copper}, {washer_type: squared,conical}

Step 4 is the simple conversion of the key-value pairs into an SQL query. We omit this trivial operation.

6.2 Question-answering

As previously mentioned, we assume that a description field is included for each fastener sample in the database. Using the description of the extracted fasteners as indicated at point 4 of the pipeline shown in Sect. 6, we perform a question-answering task as follows.

1. While the number of retrieved fasteners is higher than 50 we perform a column-splitting strategy that filters out the fasteners not important for the user - see below for details;
2. We perform extractive question-answering on the description field of the remaining fasteners and sort in decreasing order of confidence score the obtained results.

The column-splitting strategy operates as follows. Let C be the set of the retrieved fasteners. It first computes the distribution of the values for each fastener $c \in C$ along each column of the metadata. For a binary column, we will have two values associated with a distribution whereas for non-binary columns we will have the distribution of each possible value. For example, the field *TurnedStamped* (which indicates if a fastener is obtained from rotational tools like a lathe, or stamped with metal-forming techniques) is binary and each item in the database has it set to false or true. Let us suppose that for this field we have the split *true*, 35% and *false*, 65% meaning that 35% of the fasteners in C have *true* for this field and 65% have *false*. On the other hand, the field *FastenerType* (the type or fastener) is non-binary and can assume values like {Bolt}, {Nut}, {Screw}, {Washer} to mention some. Let us suppose that for this field the distribution of the values Bolt, Nut, and Washer for the fasteners in C is, respectively, 26%, 31%, and 43%. For all the other fields we operate similarly and compute the distributions as already shown. For each column, we consider the minimum value of the distributions just calculated (e.g., 35% for *TurnedStamped* and 26% for *FastenerType*). Let this set be M. To perform the column-splitting we choose the column with the highest value in M (e.g., 35% thus we choose the column {TurnedStamped}). Upon determining the optimal column, we query the user about their interest in the specific value of that column. The rationale for our column-splitting approach is to eliminate the maximum number of fasteners that do not meet the user's criteria, even if they select the value that splits the least. In our example, if the user chooses, in the worst case scenario, *false* for the field {TurnedStamped}, we could filter 35% of fasteners from C and obtain a new C'. If the size of the C' is greater or equal to 50, then the column-splitting strategy is repeated (clearly, the column {TurnedStamped} will not be a winner anymore because it will have the same value for all the elements in C').

As far as the technical details are concerned, to perform the question-answering we wrote a Python script based on the schema of the database we used. Finally, we employed *bert-large-uncased-whole-word-masking-finetuned-squad*, which is a BERT large model pre-trained using a masked language modeling objective as described in [20], and fine-tuned on the SQuAD dataset.[9]

7 Evaluation

In this section, we will illustrate the performance evaluation we have carried out for the two tasks, classification, and metadata retrieval.

[9] https://rajpurkar.github.io/SQuAD-explorer/.

Table 1 F1-measure, precision, and recall for the *k*-cross validation procedure within the synthetic dataset. Values are shown for each of the 21 classes and their average for the vision transformer and a pre-trained CNN

Name of fastener	Prec Vis %	Rec Vis %	F1 Vis %	Prec CNN %	Rec CNN %	F1 CNN %
Bolt countersunk hex inset	100	100	100	99	97	98
Bolt cylindrical head with shoulder slot inset	100	99	99	99	97	98
Bolt cylindrical head hex inset	100	100	100	100	98	99
Bolt cylindrical head slot inset	98	100	99	99	99	98
Bolt eyed	100	100	100	99	99	99
Bolt raised countersunk inset	100	99	100	98	100	99
Bolt set screw	100	100	100	98	99	99
Bolt thread insert	100	100	100	99	100	99
Bolt hexagonal	100	100	100	100	99	100
Nut castellated	100	99	100	100	98	99
Nut self-locking	99	100	100	98	100	99
Nut winged	100	100	100	100	99	100
Nut-hexagonal	100	100	100	100	99	100
Nut-grooved (slotted)	100	100	100	100	100	100
Securing circling for bore	100	100	100	99	100	100
Securing circling for shaft	100	100	100	100	99	99
Securing serrated internal	100	100	100	100	100	100
Securing tab lock	100	100	100	99	99	99
Split-pins	100	98	99	98	100	99
Rivet-blind	98	100	99	99	100	99
Washer plain	100	100	100	99	99	99
Average	100	100	100	99	99	99

Table 2 F1-measure, precision, and recall for the 210 real images (10 for each class) and their average for the vision transformer and a pre-trained CNN

Name of fastener	Prec Vis %	Rec Vis %	F1 Vis %	Prec CNN %	Rec CNN %	F1 CNN %
Bolt countersunk hex inset	80	80	80	70	70	70
Bolt cylindrical head with shoulder slot inset	83	50	62	70	70	70
Bolt cylindrical head hex inset	71	100	83	37	100	54
Bolt cylindrical head slot inset	69	90	78	62	50	56
Bolt eyed	100	90	95	88	70	78
Bolt raised countersunk inset	100	100	100	100	70	82
Bolt set screw	95	100	91	91	100	95
Bolt thread insert	100	90	95	69	90	78
Bolt hexagonal	100	80	89	100	20	33
Nut castellated	100	100	100	100	100	100
Nut self-locking	82	90	86	60	30	40
Nut winged	100	100	100	100	100	100
Nut-hexagonal	89	80	84	62	80	70
Nut-grooved (slotted)	91	100	95	53	80	64
Securing circling for bore	100	90	95	27	40	32
Securing circling for shaft	91	100	95	78	70	74
Securing serrated internal	100	100	100	100	40	57
Securing tab lock	100	100	100	46	60	52
Split-pins	83	100	91	91	100	95
Rivet-blind	100	80	89	100	60	75
Washer plain	100	90	95	67	20	31
Average	92	91	91	75	68	71

Table 3 Precision@K of the question-answering task for the proposed approach and the baseline

Method	Precision@1	Precision@3	Precision@5
Prompt Engineering	0.78	0.86	0.95
Fine-tuned Transformer	0.67	0.73	0.88

7.1 Classification

We evaluated the performance of the model in two distinct ways: (i) by only using the generated dataset with a *k*-cross validation with *k*=10; (ii) by extracting online 10 real images per fastener and testing the trained model on the resulting 210 real images. The metrics we adopted for assessing the performance of the model were F1 measure, precision, and recall. The precision, recall, and F1 values that we have obtained for the first evaluation with the proposed vision transformer and the CNN are indicated in Table 1. At each step of the *k*-cross validation, the two classifiers were trained on the same portion of the training set and tested on the same portion of the test set. As the reader may notice, the results are very high for both approaches. This outcome was expected and the reason lies within the synthetic process that has been used to create all the elements. More curious, and practical for real case scenarios, are the results of the second evaluation indicated in Table 2.

It shows the average results we obtained for each of the 21 classes of objects and the overall average for the proposed vision transformer and a pre-trained CNN on imageNet which acted as a baseline[10] (fine-tuned with weight decay equal to 0.1, batch size equal to 8 and 5 epochs).

As far as the technical details are concerned, we have used Python as the programming language and PyTorch[11] as the framework, via the Huggingface API.[12] The experiments have been run on a server with an RTX 3070 GPU. As the reader may notice, the F1 values are high, especially if we consider that we are addressing a single-label multi-class classification problem with 21 different classes. As mentioned in Sect. 3, the fasteners covered by the classification task described in this paper are often very similar to each other making the classification in 21 different classes hard. This is reflected in the performance, especially when considering the fastener *bolt cylindrical head with shoulder slot inset* and the fastener *bolt cylindrical head slot inset*. These two fasteners are very similar, and the transformer struggles to correctly classify them. This is the reason why these two fasteners have the lowest F1 scores.

7.2 Question-answering

The evaluation we carried out for the second task involved the creation of 100 questions[13] related to a set of 1000 samples previously stored in a database. The reader notices that the same fastener may clearly be present multiple times in different samples as several metadata might differ (e.g., size, material, presence of defections, color, etc.). Questions were generated by a domain expert who knew all the metadata of the fasteners. Questions were of different complexity meaning that they could contain one or more values for any field previously identified and corresponding to a database column. For example, the question:*"Can you find M10 copper bolts with a hex slot and a T-head?"*is considered simple as it includes one value for each mentioned metadata and should return the pairs:*{fasteners: bolt}, {materials: copper}, {thread_type: M10}, {slot_type: Hex}, {head_type: T-head}*.Conversely, a question like:*"Can you find all the bolts that are either M6 or M8, made of steel, and have a countersunk or button head?"*,is considered complex as it includes multiple values for some metadata and should return the pairs:*{fasteners: bolt}, {material: steel}, {thread_type: M6, M8}, {head_type: countersunk, button}*.

These questions have been run through the pipeline previously described in Sect. 6 and then through the question-answering described in Sect. 6.2.

For each question we have kept the 10 results sorted in decreasing order of confidence score. Then we defined the precision@*K* as the number of times the right result (the one looked for from the user) was found within the first *K*

[10] https://huggingface.co/microsoft/resnet-50/.
[11] https://pytorch.org/.
[12] https://huggingface.co/docs/transformers/main_classes/trainer.
[13] Publicly available at http://192.167.149.18/100questionsFasteners.txt.

candidates. We varied K in $\{1, 3, 5\}$ and in Table 3 we show the results. They have been validated by one annotator, the same domain expert who generated the 100 questions. His tasks were to read the entire collection of descriptions of the fasteners, and the first ten generated responses to each of the 100 questions. Then he had to: (i) assess whether all the key-value pairs {column name, value} were correctly extracted for each question and (ii) assess where the correct response was among the first 10 returned results.

The initial prompt resulted in 12 missing key-value pairs {column name, value}, which the annotator identified. Subsequently, prompt engineering was conducted by adjusting the prompt slightly to maximize the retrieval of pairs. After several iterations, a refined prompt was developed that successfully retrieved all key-value pairs {column name, value}, with no further missing pairs detected. This refined prompt was enhanced by incorporating examples of expected and unexpected outcomes, leveraging metadata fields that had not been recognized in earlier iterations.

We performed the same task (retrieval of {metadata field, value} using the T5-base transformer[14] and fine-tuning it with the WikiSQL dataset. The obtained SQL queries have been manually post-processed to identify the involved database fields and values. We used the semantic textual similarity[15] of Sentence Transformers[16] to match the SQL columns of the generated queries with those of our database. If the similarity score exceeded 0.6, we considered the column as a match for the underlying field. We then applied the same question-answering task to the extracted fasteners using the identical transformer utilized in our proposed in-context-learning approach.

Table 3 shows the results of the question-answering task that we carried out on top of the descriptions of the fasteners retrieved by the previous step using our in-context-learning approach and a fine-tuned transformer.

In our approach, correct answers were provided by the first response 78% of the time. Furthermore, the correct answer was found within the first 3 responses 86% of the time and within the first 5 responses 95% of the time. In cases where the correct response was not among the top 10 results, it typically appeared between the 11th and 20th response. As previously mentioned, these errors are attributed to the limitations of the fine-tuned transformers used for question-answering, which are beyond the scope of this paper for improvement. In contrast, the baseline approach performed less effectively compared to our method. This performance difference suggests that the extraction of {metadata field, value} pairs was less accurate in the baseline, likely due to using the same transformer for question-answering as our approach.

We can conclude by saying that, with the proposed in-context-learning approach, the task of extracting key-value pairs {column name, value} was successfully executed after a proper tuning of the prompt and this improves the following question-answering task.

8 Conclusions and future works

In this paper, we have discussed AI-based systems for the automatic classification of fasteners and their intelligent retrieval from a database. Two tasks have been proposed: one about image classification of fasteners and another related to the extraction of {metadata field, value} pairs of fasteners out of a database whose entries included different characteristics of fasteners. We have proposed a vision transformer trained on a collection of 2D images of fasteners to automatically perform single-label multi-class classification on 21 different classes of fasteners. The augmentation technique that we proposed starts with a few 3D samples per fastener and, by randomly varying different characteristics such as the angle from which to take an image, the distance of the camera from the object, the background against which the photo is taken, intensity, direction, and color of the illumination of the scene, generated a collection of 84.000 2D images. The second task aimed at extracting {metadata field, value} pairs from a question in natural language asking specific information about fasteners. With such a list of pairs, we could filter the database thus keeping the records with the required information. A splitting strategy has been applied to reduce the list of candidates further. Finally, we performed question-answering on the description field of the list of remaining candidates and sorted the results in decreasing order of confidence value. The evaluation that we have carried out for both tasks produced impressive results indicating that the methodologies to create the images of fasteners and to extract key-value pairs {column name, values} are effective.

While the results of our work are promising, we view them just as the tip of the iceberg. Three main objectives form the roadmap for our future activities, enabling the evolution of our concepts and ideas:

[14] https://huggingface.co/t5-baseRetraining.
[15] https://www.sbert.net/docs/usage/semantic_textual_similarity.html.
[16] https://www.sbert.net/index.html.

- Expanding the application scope of Vision Transformer to recognize a broader range of fasteners.
- Enhancing the Natural Language Processing interface correspondingly.
- Investigating methodologies to visually recognize non-standard fasteners that do not allow a description through standard attributes.

The first objective presents a considerable challenge, given the vast number of existing fasteners, often distinguishable only through small details. To train a ViT for comprehensive recognition would require an unprecedented quantity of images. A possible approach we intend to explore involves deploying multiple ViTs, each specialized in identifying individual components of fasteners, such as the head, thread, or slot. Alternatively, we may implement a hierarchical recognition method, stratifying fasteners into tiers of types, subtypes, sub-subtypes, and so forth.

The second objective, though conceptually straightforward, requires the identification of additional metadata for an extended range of fasteners - a task both intricate and time-consuming. Moreover, as the complexity of the model increases, so does the potential for erroneous responses, making validation an increasingly critical aspect.

The final objective might seem unreachable if not for the existence of universal classification methods[17] capable of categorizing any type of fasteners based on a finite set of physical features. The application of these methods is difficult also for a human and requires an understanding of mechanics: automating these methods with AI, albeit complex, would bring our methodology to another level and give a concrete contribution to the manufacturing industries.

Author contributions All the authors contributed equally to this work.

Funding Open access funding provided by Università degli Studi di Cagliari within the CRUI-CARE Agreement. We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No.3277 published on December 30, 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union - NextGenerationEU. Project Code ECS0000038 - Project Title eINS Ecosystem of Innovation for Next Generation Sardinia - CUP F53C22000430001- Grant Assignment Decree No. 1056 adopted on June 23, 2022 by the Italian Ministry of University and Research (MUR). Moreover, this research was partially funded by: H2020 Project*STAR- Novel AI technology for dynamic and unpredictable manufacturing environments (Grant Agreement 956573). H2020 Project-Mind4Machines- Connecting machines with people, process and technology (Grant Agreement 101005711).*

Data availability The dataset of 84000 fasteners images generated for this work is freely available from https://drive.google.com/file/d/1eOjZwHwMEjO8SRMDKMi5hH7nW-o88AVB/view?usp=sharing. The complete prompt used to explain ChatGPT the task to perform are publicy accessible at http://192.167.149.18/promptsFastenersGPT.txt The set of 100 questions used for the question-answering task are publicly available at http://192.167.149.18/100questionsFasteners.txt.

Declarations

Ethics approval and consent to participate Not applicable.

Competing interests Not applicable.

References

1. Abd Al Rahman M, Ebayyeh A, Mousavi A. A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry. IEEE Access. 2020;8:183192–271.
2. Kim TH, Kim HR, Cho YJ. Product inspection methodology via deep learning: an overview. Sensors. 2021;21(15):5039.
3. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, and Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[17] An example is the *Opitz classification*.

4. Mosbach M, Pimentel T, Ravfogel S, Klakow D, and Elazar Y. Few-shot fine-tuning vs. in-context learning: a fair comparison and evaluation, 2023.
5. Kumar A, Nagarkar P, Nalhe P, and Vijayakumar S. Deep learning driven natural languages text to SQL query conversion: a survey. *CoRR*, arXiv:abs/2208.04415, 2022.
6. Buscaldi D, Dessí D, Motta E, Murgia M, Osborne F, Recupero DR. Citation prediction by leveraging transformers and natural language processing heuristics. Inf Process Manag. 2024;61(1): 103583.
7. Arnab A, Dehghani M, Heigold G, Sun C, Lucic M, Schmid C. Vivit: A video vision transformer. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, IEEE Computer Society: Los Alamitos, CA, USA, 2021. pp. 6816–6826
8. Han K, Wang Y, Chen H, Chen X, Guo J, Liu Z, Tang Y, Xiao A, Xu C, Xu Y, Yang Z, Zhang Y, Tao D. A survey on vision transformer. IEEE Trans Pattern Anal Mach Intell. 2023;45(01):87–110.
9. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, and Guo B. Swin transformer: hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International conference on computer vision (ICCV)*, 2021;9992–10002.
10. Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, and Guo B. Swin transformer: hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International conference on computer vision (ICCV)*, 2021;9992–10002.
11. Yin H, Vahdat A, Alvarez JM, Mallya A, Kautz J, and Molchanov P. A-vit: adaptive tokens for efficient vision transformer. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, IEEE Computer Society, Los Alamitos, CA, USA, 2022. pp. 10799–10808
12. Yu T, Zhang R, Yang K, Yasunaga M, Wang D, Li Z, Ma J, Li I, Yao Q, Roman S, Zhang Z, and Radev D. Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, Association for Computational Linguistics: Brussels, Belgium, 2018. pp. 3911–21
13. Deng N, Chen Y, and Zhang Y. Recent advances in text-to-SQL: a survey of what we have and what we expect. In *Proceedings of the 29th International conference on computational linguistics*, International Committee on Computational Linguistics. Gyeongju, Republic of Korea, 2022. pp. 2166–2187
14. Yi L, Gelei D, Xu Z, Yuekang L, Yaowen Z, Ying Z , Lida Z, Tianwei Z and Liu Y. Jailbreaking chatgpt via prompt engineering: an empirical study; 2023.
15. White J, Fu Q, Hays S, Sandborn M, Olea C, Gilbert H, Elnashar A, Spencer-Smith J, and Schmidt DC. A prompt pattern catalog to enhance prompt engineering with chatgpt, 2023.
16. Mumuni A, Mumuni F. Data augmentation: a comprehensive survey of modern approaches. Array. 2022;16:100258.
17. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. J Big Data. 2019;6(1):60.
18. Alomar K, Aysel HI, Cai X. Data augmentation in classification and segmentation: a survey and new strategies. J Imag. 2023;9(2):46.
19. Dosovitskiy A, Beyer L, Kolesnikov Ar, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR. 2021.
20. Devlin J, Chang MW, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, arXiv:abs/1810.04805. 2018.