Raw Material Selection for Object Construction

Jason Perlow[1], Benjamin Rosman[1,2], Bradley Hayes[3] and Pravesh Ranchod[1]

***Abstract*—An important step in the construction of novel objects is the ability to recognise combinations of raw materials which are likely to be useful. We aim to exploit the intuition that the visual characteristics of candidate raw materials provide useful cues to their potential combinations. Toward this end, we present a Siamese neural network based model that is able to recognise unseen raw materials present in objects given a list of candidate material images. We demonstrate the utility and efficacy of our model within two domains. The first being a proof-of-concept within Minecraft where we predict the combinations of objects that will result in a target object. The second, more realistic domain, uses the ShapeNet 3D model dataset where we attempt to recover the materials present in a model. We empirically demonstrate that our model is able to learn from a subset of object material pairs and generalise to unseen objects, materials, texture packs. Under these conditions of high visual variation, we show that our model outperforms chance and baseline methods.**

I. INTRODUCTION

For a construction agent to select the raw materials necessary to construct a novel object, it may need to recognise classes of raw materials present in the object that it has not seen before. To achieve this, the agent would need to perform recognition beyond fixed classes and transfer knowledge from familiar raw materials and object classes to novel raw material and object classes.

As a step towards creating such an agent, we propose a model that performs recognition by making use of the general visual relationship between a given material and an object it needs to construct. The model learns the general visual relation by being given a subset of objects and materials which allows the model to compare unseen objects and materials without additional training.

For example, consider the case shown in Figure 1 where previous observations of the material on the top left can be used to create the tool on the top right. By analogy, the agent can then predict that the materials on the bottom left are necessary to create the tool on the bottom right. In such a case, the agent has not seen the target tool nor one of the materials. By learning to recognise the similarities between the first tool and its material, it can use this similarity knowledge to select appropriate unseen materials for unseen tools.

[1]are with the School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa. Corresponding author: `jason.perlow@students.wits.ac.za`

[2]is with the Council for Scientific and Industrial Research, Pretoria, South Africa.

[3]is with the Department of Computer Science, University of Colorado Boulder, USA.

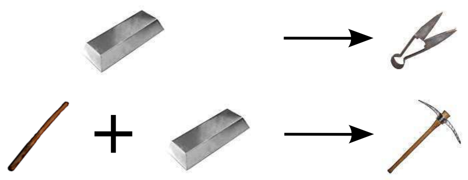

Fig. 1. (Top) Source domain - Construction of iron shears. (Bottom) Target domain - Constructing an iron pickaxe from raw materials.

A. Visual Similarity

We propose a model inspired by the recognition of visual similarities between required materials and a desired object that an agent needs to construct. This allows an agent to construct unseen objects more quickly than trying all possible materials available, by solely using features based on their visual appearance. In particular, we present a method for an agent to recognise the required unseen raw material images and link them to corresponding novel object images. This capability provides an agent with an increased degree of resourcefulness in unseen environments [1].

B. Domains

We demonstrate the efficacy of visual similarities to perform material selection in two domains. The first involves tool construction in the sandbox adventure game and reinforcement learning domain Minecraft [2]. The second domain is that of material recognition performed on 3D furniture models from the ShapeNet dataset [3]. The ShapeNet domain aims to demonstrate the applicability of our method beyond an adventure game and show that it could, in the future, form part of a real-world model for an agent capable of object construction.

In the Minecraft domain we test our model's ability to find the visual features necessary to perform tool construction. We use the crafting (inventory item combination) mechanic from Minecraft. In Minecraft, raw materials can be mined from the environment and combined or crafted into food, tools and other objects that can be used in the game. We focus our attention only on selecting necessary ingredients and ignore the position and quantity of materials on the in-game crafting grid or table.

As an example of determining the recipe of the tool by finding the most plausible raw material combination, consider possible recipes in Figure 2. Even though both the tool and material are unseen, from inspection of Figure 2, a human without any Minecraft experience can see that the third combination is most likely. It is precisely this intuition derived from visual features (such as shape and colour) that we successfully model in this paper.

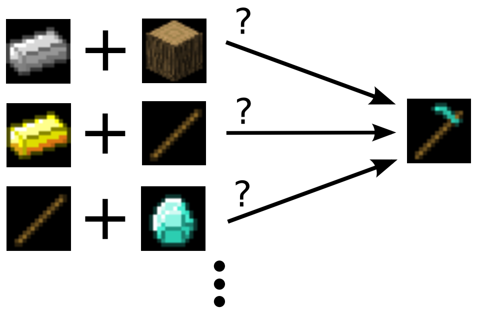

Fig. 2. Possible inventory item ingredients for crafting a diamond hoe. Note the visual similarity of the third row (the correct pair) and the tool. This intuition inspires our model. *Best viewed in colour.*

To further motivate the adaptivity of our method to visual variation we also consider generalisation to tool images of unseen *visual style*. Since the chosen domain of Minecraft is restricted in the visual variety of available items, we mitigate the limitations of the default dataset, the vanilla texture pack, by selecting a variety of additional texture packs. For example we use the LB Photorealism texture pack (see Figures 1 and 5), which has images with a more realistic appearance compared to the vanilla texture pack (see Figure 2).

Building on the intuition of the Minecraft domain, we exploit the same visual similarities between objects and their constituent materials in the ShapeNet domain. As shown in Figure 3, in the ShapeNet domain we aim to determine the presence of raw materials in a given 3D object model by matching a list of 3D model textures with rendered 3D models that contain exactly one of those textures. This domain therefore assumes that textures correspond to raw materials such as fabric, light and dark wood (shown in Figure 3). Here we test the ability of our model to generalise to unseen materials once trained on a constrained subset of material object pairs.

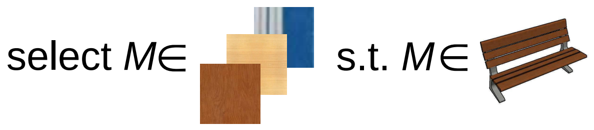

Fig. 3. Possible materials for building a bench. *Best viewed in colour.*

Our primary contribution in this work is a Siamese neural network-based model that predicts the probability of a construction/crafting success given component raw material images. In the Minecraft domain, these are M_1 and M_2 from an inventory $\mathcal{M}$. More formally, for a particular target or goal object $G \in \mathcal{G}$, the model generates a series of mappings, which when ranked, approximate the crafting procedure that maps two inventory items to a goal item:

$$\mathcal{M}^2 \rightarrow \mathcal{G} \tag{1}$$

To find the correct crafting policy π given G, we invert the crafting model to yield

$$\pi : \mathcal{G} \rightarrow \mathcal{M}^2. \tag{2}$$

However, in the ShapeNet domain we wish to find a single material and thus consider $\mathcal{M}$ instead of $\mathcal{M}^2$ in Equations (1) and (2).

We show that our model successfully extracts visual features from the appropriate raw materials that best match the visual features of the object to be constructed, allowing for more rapid object production than competing methods.

To the best of our knowledge, this is the first such model using visual similarity transfer to accelerate vision-based object construction from component parts, and in doing so we provide a benchmark for future work to compare against within Minecraft and ShapeNet domains.

II. BACKGROUND

Our model is inspired by Siamese neural networks, a class of neural network that includes multiple identical subnetworks in its architecture, particularly well suited for finding similarity. Siamese neural networks include popular network architectures such as the convolutional neural network (CNN) [4], which we utilize in our work.

A. Siamese Networks

Siamese networks have multiple neural network branches with shared weights that produce vectors that are fed into a similarity metric [5]. A common similarity metric is the cosine similarity $cos(\cdot,\cdot)$ which is proportional to the angle between two vectors. Since the neural network branches are typically CNNs, the output y of the whole model can be stated as

$$y = cos(CNN_w(x_1), CNN_w(x_2)), \tag{3}$$

where x_1 and x_2 are images and w are weights which are shared between the two CNN branches. Conceptually, a Siamese network therefore extracts the same features from two images and compares them to determine their visual similarity.

B. Training

Since the Siamese network outputs a scalar similarity, training reduces to a regression problem. We update weights such that a loss function is minimised between the ground truth $\bar{y}$ and output y of the network. We use the cosine embedding loss as defined in the Torch machine learning library [6], given by

$$loss(y, \bar{y}) = \begin{cases} 1 - y, & \bar{y} = 1 \\ max(0, y - m), & \bar{y} = -1 \end{cases} \tag{4}$$

where m is a margin.

III. METHODS

To predict the materials needed to create objects we introduce two models. The first model is intended to be a simple performance baseline and conceptual starting point for the more complex second model. Both approaches require raw materials and desired object images as inputs, and both output the correlation of input-output pairs to indicate a successful or unsuccessful construction.

Our ShapeNet model is a simplification of our Minecraft model in that it only detects single materials and therefore has one branch where the Minecraft models have two.

A. Pixel Correlation

As a reasonable baseline we use the intuition of correlating the relative pixels in each image. To this end, we propose a model which sums the raw pixels of inventory item sprites M_1 and M_2 and then measures the cosine similarity with the sprite image of the goal tool we wish to construct, G. This addition network (AddNet) is therefore stated formally as

$$AddNet(M_1, M_2, G) = cos(M_1 + M_2, G). \tag{5}$$

To find the best images to use to construct a particular goal, we execute the model on all possible policies, being unique pairs from the inventory $\mathcal{M}$. To find the combination of items that have the highest similarity with the tool we want to construct, correlations are ranked and crafting attempts are made until a correct recipe is found. The aim of the model is therefore to reduce the number of crafting queries and thus crafting attempts needed to produce a tool. The optimal policy predicted by the model is therefore

$$\pi(G) = \arg\max_{i,j} AddNet(M_i, M_j, G). \tag{6}$$

Item pairs can include two of the same image for the situation where a recipe only has a single ingredient. AddNet is commutative, which models the fact that order does not matter in crafting.

B. Visual Similarity

We now build on from the intuition of AddNet, by extracting visual similarities from the raw images. This gives rise to the full Minecraft model: an extension of AddNet using convolutional branches with shared weights. The convolutional layers and similarity metric of the model bear a strong resemblance to Siamese networks. However, as shown in the architecture diagram in Figure 4, the outputs of the two input convolutional layers are added before the similarity metric is used. We use

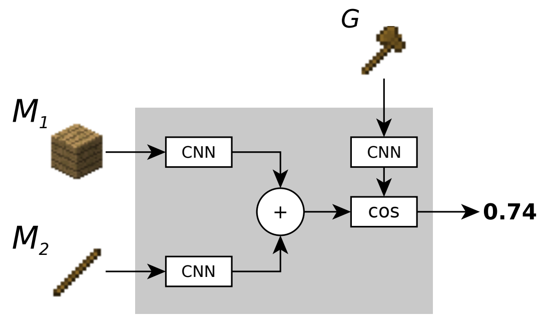

Fig. 4. Minecraft model architecture (in grey) with example input and output.

CNN branches to map the sprites onto a Euclidean space so they can be more easily added and correlated as vectors. This mapping is interpreted as visual feature extraction and therefore takes the sum of the visual features from the ingredients and correlates them with visual features of the goal.

Since we want the model to output composition correlations, model weights are optimised such that it outputs 1 when ingredients are predicted to successfully craft and -1 when they do not. When training our Minecraft model, the sum of the embedded ingredient vectors are expected to share similarities with the embedded goal item and are therefore regressed to 1 whereas recipes that are dissimilar are regressed to -1.

As with AddNet, all possible unique pairs are fed into the model to produce crafting correlations. Crafting attempts are then made based on ranked crafting correlations.

In the case of ShapeNet our model instead becomes a simple Siamese network that merely correlates visual features extracted from candidate materials, using a CNN, with the object that needs to be constructed.

C. Justification of Methods

Both AddNet and CraftNet use the cosine similarity metric. A motivation for this metric is that it is invariant to the magnitude of both vectors. This is important in the case where a tool needs two of the same ingredient. Here, the cosine between the sum of two identical images $M_1 + M_1$ and another image M_2 is the same as the cosine between a single image M_1 and the other image M_2. Since we represent single ingredient recipes as $M_1 + M_1$ where M_1 is the ingredient, this cosine property models the intuition that crafting both one or two ingredients should have the same result.

Empirical tests have determined that a Siamese neural network outperforms a monolithic image classification type CNN on visual analogy problems [7]. Since our model is inspired by Siamese neural networks and deals with visual analogy problems, we draw inspiration from these results. Furthermore, by having separate branches for each input, the model does not prematurely mix data from separate images until the addition and cosine operations are performed. This is in contrast to a monolithic CNN model which would add images from different materials in early convolutional layers.

A key assumption of our full model is that extracted visual features from the materials can be added and then correlated with the features from the final product. We therefore focus our attention on object construction where, from inspection, visual similarities between objects and their respective materials do appear to exist.

IV. EXPERIMENTS

For Minecraft, we evaluate both of our models on a total of seven experiments, using different sets of raw materials, goal tools, and sprite texture packs. The experiments use increasingly sparse data. In particular, experiments 1–3 each exclude individual categories of tools, materials and texture packs (representing different styles) from training data respectively. Training data is further reduced in experiments 4–7, where we exclude combinations of two of these unseen categories. In addition to decreasing training data, we also change the orientation of test images compared to training images. The experiments are therefore constructed to test the robustness of our model against increasing degrees of unseen data and visual variation. In all experiments baseline methods are benchmarked against chance which uniformly samples from available remaining materials.

A. *Minecraft Data*

Out of the many inventory items in Minecraft, we focus on experiments that craft a subset of all items. Namely, we focus on tools made from wood, stone, iron, gold and diamond[1]. The ingredients that can be used to make these tools include refined materials and their respective ores. The dataset then has a total of 26 tools and 11 ingredients. Tools and ingredients in vanilla Minecraft are represented by 32×32 pixel colour images or sprites. For the purpose of our experiments we resized all sprites to 32×32 pixel images and pixels were scaled between 0 and 1, but were not adjusted for brightness or contrast. Sprite samples from different texture packs are shown in Figure 5.

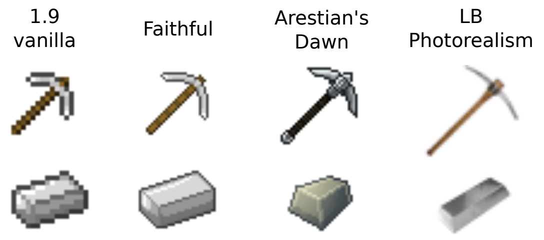

Fig. 5. Examples from the texture packs used as training data in experiments. In experiments 4, 5 and 7, two of the texture packs were unseen.

B. *Minecraft Model*

In the seven experiments, CraftNet is a two layer convolutional and one dense layer based model with $\sim$ 200k parameters trained on different tool sets as shown in Table I. For example in experiment 6, we train the model on sprites from all four texture packs which are shears together with wood, iron and stone hoes, axes and swords. We then test AddNet and CraftNet on the remaining tools and materials from all texture packs to determine the generalisation of the model.

In addition to tool and material constraints, in experiments 3, 5 and 7, we perform experiments similar to the other four experiments but train the model on two texture packs (vanilla 1.9 Minecraft texture pack and Arestian's Dawn) and test it on the remaining two (Faithful and LB Photorealism). For example, in experiment 3, we train on all tools and materials from these two texture packs and test the model on the remaining two texture packs. Sample sprites from all four texture packs are shown in Figure 5. When a model makes a prediction input images are drawn from the same texture pack so visual similarities can be detected.

Since crafting can be successful or unsuccessful, we used both positive and negative training pairs. In experiments where

TABLE I
TOOL MATERIAL TYPES USED AS TRAINING DATA IN THE FIRST 6 EXPERIMENTS. EXPERIMENT 7 USES THE SAME TOOLS AS EXPERIMENT 6, BUT IS ONLY TRAINED ON TWO TEXTURE PACKS, SIMILAR TO EXPERIMENTS 4 AND 5.

	Wood	Stone	Iron	Gold	Diamond
Axe	1–6	1–6	1–6	1,3	1,3
Hoe	1–6	1–6	1–6	1,3	1,3
Sword	1–6	1–6	1–6	1,3	1,3
Pickaxe	2–4	2–4	2–4	3,6	3,6
Shovel	2–4	2–4	2–4	3,6	3,6
Shears	none	none	1–6	none	none

materials are excluded, excluded materials and the ingredients used to make them are also excluded from negative training pairs. Many more item combinations cannot be crafted than can be crafted. To remedy this, in each training round we randomly select a different subset of 10 items from the oversampled class so the model is both exposed to the whole undersampled class while not being overwhelmingly exposed to data from the oversampled class.

Recipe ground truths were collected from the Minecraft Wiki[2]. The recipes can consist of mappings between one or two inventory items and a single tool.

A key goal of the proposed model is to generalise material recommendations to novel tools where the model has not seen materials, tool types or graphical styles. To quantify generality we measure the average number of failed attempts taken to craft all unseen tools. This average is then normalised by dividing it by 64, the maximum possible failed attempts, yielding a mean recall metric. The results of the 7 experiments are summarised in Table II and in particular, the results of experiment 7 are shown in Figure 6. In Figure 6 the proportion of unseen test tools correctly crafted is measured at k, a threshold of the number of failed crafting attempts, namely Recall at Top-k [7]. For this Figure, results for the same tool from different texture packs are averaged together while the chance model is averaged over 100 trials and has error bars of one standard deviation.

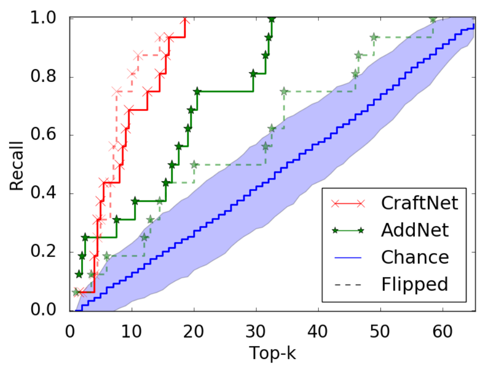

Fig. 6. A comparison of the number of crafting attempts needed to craft test tools between chance, the proposed model and the baseline in experiment 7. "Flipped" refers to the more difficult case of some of the sprites being flipped along the vertical axis.

[1]These are defined in http://minecraft.gamepedia.com/Tools.

[2]http://minecraft.gamepedia.com/

TABLE II

MEAN RECALL OF CRAFTNET COMPARED TO THE ADDNET BASELINE AND CHANCE. EXPERIMENT RESULTS FOR FLIPPED SPRITES ARE IN PARENTHESIS. STANDARD DEVIATION ERRORS ARE SHOWN FOR CHANCE EXPERIMENTS. BEST RESULTS ARE IN BOLD.

Unseen Data	AddNet	CraftNet	Chance
1. Tools	0.674 (0.395)	**0.826 (0.806)**	0.481±0.296 (0.542±0.315)
2. Materials	0.612 (0.301)	**0.777 (0.717)**	0.572±0.289 (0.507±0.272)
3. Texture packs	0.742 (0.531)	**0.855 (0.848)**	0.529±0.260 (0.432±0.272)
4. Tools & texture packs	0.691 (0.506)	**0.855 (0.852)**	0.430±0.332 (0.565±0.315)
5. Materials & texture packs	0.775 (0.598)	**0.851 (0.832)**	0.490±0.292 (0.581±0.294)
6. Materials & tools	0.655 (0.441)	**0.812 (0.817)**	0.555±0.268 (0.568±0.308)
7. Materials, tools, & texture packs	0.737 (0.557)	**0.860 (0.880)**	0.501±0.285 (0.484±0.287)

In all experiments, CraftNet performed on average better than AddNet and chance. However, as shown in Figure 6, in the particular case of experiment 7, in terms of the steps to craft tools, CraftNet performed comparably to AddNet for some tools. That is, for easy tools where there is a high degree of visual overlap the baseline and the proposed model are comparable. An example of this are wooden tools where the wooden planks sprite overlaps with the head of the tools and will therefore correlate well.

Even when some of the input sprites are flipped along their vertical axis, CraftNet does not decrease in performance and still performs better than chance despite having not being trained on flipped sprites (with the exception of shovels in the LB Photorealism texture pack). This is in sharp contrast to AddNet which under the same conditions is comparable with chance when attempting to craft any item. A possible reason for this is that, as shown in Figure 7, flipped sprites often do not overlap with the goal item thus demonstrating the fragility of AddNet. These results further indicate that the proposed CraftNet model is robust against visual variation compared to the AddNet baseline which fails under simple image perturbations.

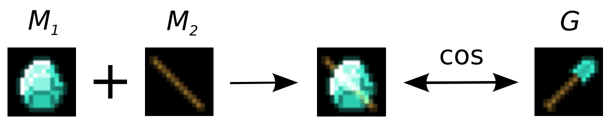

Fig. 7. An example of poorly overlapping flipped ingredient sprites resulting in poor AddNet performance.

C. ShapeNet Data

From the ShapeNetCore dataset we select categories of objects that an assembly agent could plausibly construct from raw materials, such as chairs, tables and cabinets while excluding categories like aeroplanes, bottles and mugs. We also balance the data such that there no more than 500 samples per category. We render objects from all eight cardinal views using the provided ShapeNet viewer[3] and resize rendered images and textures to 128×128 images. In an attempt to model material selection in objects, we focus our attention on objects that are largely covered by textures since the appearance of most 3D models are determined by both textures and internal mesh data. As shown in Figure 8, after rendering, we filter out such models by swapping out textures for first blue then red textures and comparing renderings. When there is a large Euclidean distance between images, the appearance of the 3D model is likely largely determined by textures and is therefore included in our experiments. This filtering process yields a total of 1008 models.

[3]http://github.com/ShapeNet/shapenet-viewer

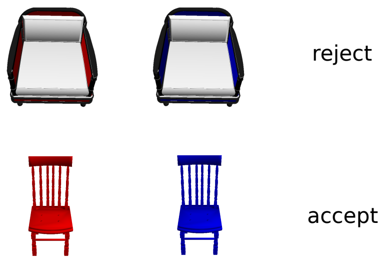

Fig. 8. Change of textures in models to measure the impact of textures on visual appearance. (Top) Minimal change - Reject. (Bottom) Significant change - Accept. *Best viewed in colour.*

D. ShapeNet Model

Similar to the Minecraft model, each branch in the ShapeNet model has two convolutional layers followed by one dense layer which together have a total of $\sim$ 250k parameters. Similar to Minecraft we hold back 20% of all materials and test on the remaining 80%. As an example we hold back a wood texture during training and require the model to recognise wood in an object. We also randomise weights as an additional benchmark to see if our model does indeed transfer knowledge from seen classes to unseen classes. We also compare our model against colour histogram features to see if our model learns additional features beyond simple colour matching. The results of these experiments are measured with mean recall are shown in Table III. The trained model performed better than chance, our baseline model, a colour histogram baseline and a full model with randomised weights.

V. RELATED WORK

Related to our proposed model is work that uses affordances, visual analogies and neural networks in a planning or classification setting.

Humans use affordances that are visual cues encoded in the environment which afford a particular action and thus enable the person to interact with unfamiliar objects [8]. This relates to the presented work which uses visual features since affordances are a type of visual feature used in a planning context. Affordances have been used in Minecraft to prune possible actions in a similar way to how we use visual similarities to prune the list of possible recipes [9].

TABLE III
MEAN RECALL OF THE TRAINED SHAPENET MODEL COMPARED TO RANDOMLY INITIALISED WEIGHTS, OUR CORRELATION BASELINE AND CHANCE. STANDARD DEVIATION ERRORS ARE SHOWN FOR RANDOMISED EXPERIMENTS. BEST RESULTS ARE IN BOLD.

Unseen Data	Baseline	Colour Histogram	Random Weights	Model	Chance
Materials	0.522	0.525	0.559±0.256	**0.751**	0.497±0.332

Our proposed framework also has similarities with affordance based tool and grasping research. An example is using hard-coded affordance features to cluster the grasping types of 3D tool models [10]. This allows for the detection of grasping configurations of unseen tools, which is similar to the proposed method which uses visual similarities to construct objects. Other examples include learning how to use affordances such that a robotic arm can interact with novel objects using a real-world robotic arm [11] and a simulated one [12].

Another body of research on which the framework of this paper is based is that of visual analogies [13]. Sadeghi *et al.* [7] use a Siamese network to predict visual relations of objects in the following form: given a visual mapping $A \rightarrow B$, find x in the relationship $C \rightarrow x$. This is related to the presented material selection problem in that we want to find mappings between combinations of materials and objects given prior analogous mappings. Inspiration for image addition in our Minecraft model and baseline comes from Reed *et al.* [14], who performed image additions using an additive CNN with a decoder to predict the visual appearance of video game character sprites in unseen orientations.

Neural network based models are also used in the field of material recognition. Material recognition is related to the material selection in the presented work. However, work in material recognition has mainly focused on recognition with fixed material classes [15]. This is in contrast to the presented work which recognises materials by being shown an example and is therefore not restricted to fixed classes. Unlike material recognition we demonstrate the ability to transfer to new materials. Furthermore unlike previous material recognition work, we apply our method to full isolated objects and not full scenes or patches within a scene [16].

VI. CONCLUSION

We have shown that our proposed models outperform chance as well as an image correlation baseline in two domains consisting of varying object types and visual appearance. Since the baseline is more sensitive to simple perturbations than our proposed models, the addition of CNN-based feature extraction layers in the presented models are necessary.

Since the LB Photorealism texture pack and the ShapeNet rendered models have realistic images, they hint at the possibility of a real-world object construction agent. However, even in these datasets, objects are still not completely realistic. They sometimes have intentionally exaggerated colour (e.g., bright blue for diamond tools and materials). Also, even in realistic sprites and rendered 3D models, there is no background clutter which would exist in real domains.

The combinatorial nature of the material search algorithm means that the model needs to operate on many inputs, $O(n^2)$, before ranking can take place. To remedy this we could use a two-level ranking process inspired by the Learning to Rank literature [17]. Namely, a coarse filter would first rank groups of materials, and then would apply the proposed method to refine the results within top ranking groups.

REFERENCES

[1] J. T. Balint and J. M. Allbeck, "Macgyver virtual agents: using ontologies and hierarchies for resourceful virtual human decision-making," in *Proceedings of the 2013 International Conference on Autonomous agents and multi-agent systems*, 2013, pp. 1153–1154.

[2] K. Aluru, S. Tellex, J. Oberlin, and J. MacGlashan, "Minecraft as an experimental world for ai in robotics," in *AAAI Fall Symposium*, 2015.

[3] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[5] J. Bromleyh, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 04, pp. 669–688, 1993.

[6] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," Idiap, Tech. Rep., 2002.

[7] F. Sadeghi, C. L. Zitnick, and A. Farhadi, "Visalogy: Answering visual analogy questions," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 1882–1890. [Online]. Available: http://papers.nips.cc/paper/5777-visalogy-answering-visual-analogy-questions.pdf

[8] J. J. Gibson, *The Ecological Approach to Visual Perception*. Psychology Press, 1986.

[9] G. Barth-Maron, D. Abel, J. MacGlashan, and S. Tellex, "Affordances as transferable knowledge for planning agents," in *2014 AAAI Fall Symposium Series*, 2014.

[10] T. Mar, V. Tikhanoff, G. Metta, and L. Natale, "Multi-model approach based on 3D functional features for tool affordance learning in robotics," *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 482–489, 2015.

[11] B. Ridge, D. Skočaj, and A. Leonardis, "Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 5047–5054.

[12] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *7th IEEE International Conference on Development and Learning*, 2008, pp. 91–96.

[13] B. M. Stafford, *Visual analogy: Consciousness as the art of connecting*. MIT Press, 2001.

[14] S. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Advances in Neural Information Processing Systems*, 2015, pp. 1252–1260.

[15] C. Liu, L. Sharan, E. H. Adelson, and R. Rosenholtz, "Exploring features in a beyesian framework for material recognition," in *Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 239–246.

[16] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] A. Z. Broder, D. Carmel, M. Herscovici, A. Soffer, and J. Zien, "Efficient query evaluation using a two-level retrieval process," in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003, pp. 426–434.