Visually Identifying Rank

David F. Fouhey, *Mathematicians Hate Him!*
Daniel Maturana, *Random Forester* Rufus von Woofles, *Good Boy*

Abstract—The visual estimation of the rank of a matrix has eluded researchers across a myriad of disciplines many years. In this paper, we demonstrate the successful visual estimation of a matrix's rank by treating it as a classification problem. When tested on a dataset of tens-of-thousands of colormapped matrices of varying ranks, we not only achieve state-of-the-art performance, but also distressingly high performance on an absolute basis.

Index Terms—perceptual organization; vitamin and rank deficiencies; egalitarianism in the positive-semi-definite cone; PAC bounds for SVDs; class-conscious norms

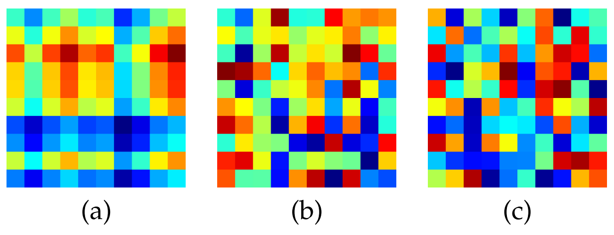


Fig. 1. What are the ranks of these matrices? Which ones are rank-deficient? In this paper, we investigate how one can guesstimate the rank of a matrix from visual features alone. See footnote on page 2 for answer.

1 Introduction

Consider Figure 1(b): what is the rank of the matrix? Most people are confused. Some might hazard a guess. A select collection of professors might say "3." The mystery of how professors can visually estimate the rank of matrices from as little as a brief glance at a jet-colormap rendering has puzzled researchers in neuroscience, philosophy, mathematics, and computer science for decades.

The rank of a matrix $\mathbf{M}$ reveals a great deal. By definition, it tells us how many linearly independent columns the matrix has; surprisingly, it also tells us how many linearly independent rows the matrix has, and if that does not get you excited, I do not know what will. If we think of $\mathbf{M}$ as an operator, the rank tells us about the dimensionality of its output, and thus for a square matrix, whether $\mathbf{M}$ is invertible.

In this paper, we show how to identify the rank of a matrix from an image alone. In contrast to past work on guaranteed solutions to matrix rank computation that require access to the matrix, our work gives guarantee-free solutions that can operate on only an colormapped version of a matrix. By treating matrix rank as an image classification problem, we are able to consistently achieve distressingly high performance – $\approx$ 40% accuracy on 10-way classification; $\approx$ 80% accuracy on rank-deficient/not-rank-deficient binary classification. In subsequent experiments we show the following: 1) Our method can identify what matrices seem low rank, and why; 2) Our method is easily extended to structured prediction; 3) That activations of our network can be even used as a feature for semantic image classification with non-embarrassing performance (20.9% on Caltech 101 with 15 samples).

• *All authors are with The Robotics Institute, Carnegie Mellon University.*
Send us fan mail at:
Neurotic Computing Institute c/o D. Fouhey, A.B. M.S.
EDSH 212
5000 Forbes Avenue
Pittsburgh, PA 15213

2 Related Work

In this work, we tackle two problems concerning a square matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$. The first is a binary classification problem: is $\mathbf{M}$ full-rank? The second is a k-way classification problem: what is the rank of $\mathbf{M}$?

Many approaches exist for solving both problems. In the binary case, for instance, note that a square matrix is full rank if and only if its determinant is non-zero. This leads to a straight-forward way to check for rank deficiency. Similarly, if we let $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{M}$ be the singular value decomposition of $\mathbf{M}$ (i.e., $\mathbf{\Sigma}_{i,i} = \sigma_i$ where σ_i is the ith singular value of $\mathbf{M}$), then the rank is the number of non-zero singular values. This permits checking not only for rank deficiency but also calculating the rank.

While these sorts of approaches enable the accurate solution of both questions, they (a) require access to the matrix itself (as opposed to a screen capture or printout) and (b) have time complexity greater than $O(n^2)$. SVD computation has complexity $O(n^3)$ and determinant calculation is $O(n^3)$ with Bareiss [1] or $O(n^{2.807})$ with Bunch and Hopcroft [2]. Our work aims to fix these gaps.

Our method requires only access to a visual representation of the matrix, and thus answers the purely

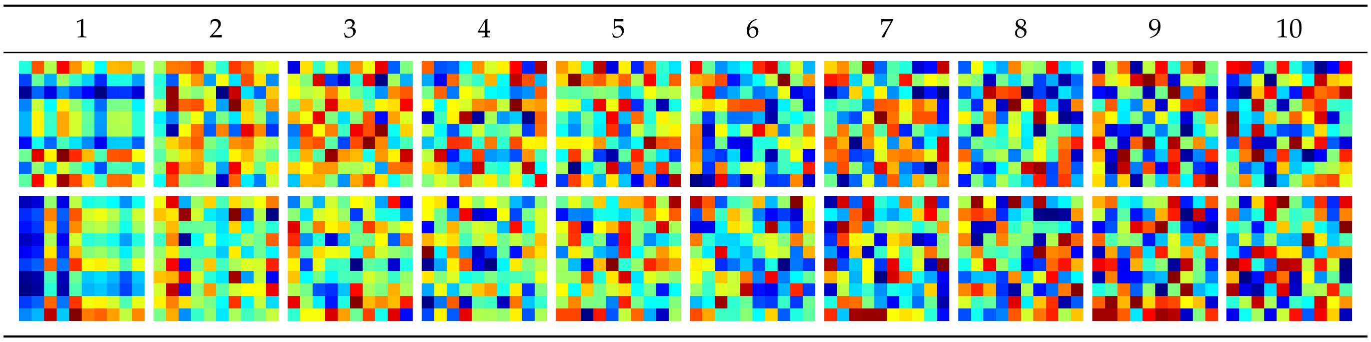


Fig. 2. Examples of matrices of various ranks. Top row: random instances; Bottom row: archtypical examples of each rank, determined by the most confident classification examples from a pool of 1,000 matrices according to a classifier.

visual way of computing rank as opposed to the *mechanical* way of computing rank. When a professor says a matrix looks rank-deficient, she is probably not doing an SVD, but instead using some visual smell-test (akin to the notion of direct perception as proposed by psychologist J.J. Gibson [3]); we seek to emulate this astounding ability.

Our method is $O(n^2)$. Feature extraction is only dependent on the number of pixels for all methods and thus $O(n^2)$. If we are doing binary classification, it is $O(1)$ and thus the method serves as a guarantee-free quadratic-time rank-deficiency test. If we do n-way classification, it depends on the method, but is arguably $O(1)$ for random forests, which most of our methods use.

We acknowledge that our work gives no guarantees, but computer vision has a long history of extraordinarily successful algorithms that may not always be right. The Random Sample Consensus algorithm [4], for instance, can give no guarantees about its performance on any individual problem instance. However, it works extraordinarily well in practice, and the authors have taken it all the way to the proverbial citation bank with a well-deserved ≈10,000 citations. Similarly effective methods include Simulated Annealing, Iterative Conditional Modes (ICM) [5], and many others. Our method may not be as successful, but as presented in Table 1, our method is surprisingly effective and has a certain je ne sais quoi.

Our approach of neural-network based approaches to linear algebra is not itself novel. However, previous approaches (e.g., [6], [7], [8], among many others) require direct access to the matrix itself. Our approach, on the other hand, only gets access to an colormap of the matrix.

The natural complement to the thematically-related related work section (as above) is the alphabetically-related related work section introduced in Fouhey and Maturana's seminal work on celebrity-themed learning [9]. In this paper, we extend this to a word-based related work section. A highly related technique is the rank transform proposed by Zabih and Woodfill [10]. This method replaces each pixel by its "neighborhood rank" to achieve invariance to monotonic illumination differences. We employ a related technique, Local Binary Patterns [11] to extract features for our classifiers. Also related are learning-to-rank algorithms such as support vector ranking [12].

1. Answers first page quiz: a – rank 1; b – rank 3; c – rank 10.

TABLE 1
A comparison of ways to check whether a matrix is rank deficient. We evaluate methods on their time complexity, their success rate, whether there is an easy extension to n-way rank calculation, and their Je Ne Sais Quoi.

Method Name	Complexity (Time)	Success Rate	Multi-class Extension	Je ne sais quoi
SVD	$O(n^3)$	**100%**	**Yes**	Minimal
Det.	$O(n^{2.807})$	**100%**	No	Kinda
CNN	$\mathbf{O(n^2)}$	78.6%	**Yes**	**Tons**

3 Technical Approach

We now introduce the method in simple English to illustrate its simplicity. We take a matrix, visualize it as a picture (like a `.png`), and feed it into a standard image classification pipeline. More formally, we create a fixed-length feature representation ϕ of the image, and learn a mapping f that maps the representation to a set of discrete classes. For instance, we might extract standard image features like SIFT as ϕ, and apply a standard technique like Random Forests or SVM to learn an f. Similarly, we might train a convolutional neural network (CNN) to predict the rank, serving as both ϕ and f. This classifier is simply trained on a collection of random matrices. We note that one elegant aspect of our method is that rank-deficiency and classification are encapsulated in the same learning formulation.

3.1 Features and Learning Method

In this paper, we apply standard image classification machinery by substituting in various standard fea-

TABLE 2
Quantitative results on visual rank problems. Our paradigm of rank-prediction works surprisingly well across a myriad of features and learning methods.

Learning method f	Random Forest+Engineered				RF + Pretrained		Scratch CNN	Chance
Feature map ϕ	*All Eng.*	*Gray SIFT BoW*	*Color SIFT BoW*	*LBP BoW*	*pool*$_5$	*fc*$_7$	*Raw Pixels*	
10-way Rank	38.1%	32.5%	36.5%	31.0%	33.7%	34.9%	**43.5%**	10%
Rank Deficiency	76.4%	73.1%	75.3%	73.9%	75.0%	76.3%	**78.6%**	50%

tures and learning methods for ϕ and f.

Shallow Learner + Features: *Features (ϕ):* The first feature type we use is standard hand-engineered features in the form of a bag-of-words (i.e., histogram) representation over dense SIFT [13] and Local Binary Patterns (LBP) [11]. To quantize SIFT, we build a codebook with k-means; each extracted SIFT feature is represented by the nearest cluster center (i.e., hard assignment). Thus, each image is mapped to one or more histograms of codewords; we concatenate histograms when using multiple representations. We also experiment with using the responses of a standard convolutional neural network – Alexnet [14] – pre-trained on the Imagenet dataset [15]. We use the standard pool$_5$ and fc$_7$ features.

Learning Methods (f): We work with random forests [16] although our method is entirely generic. In this method, an ensemble of decision trees is trained independently. During learning, splitting occurs on a random subset of features and occurs until a minimum number of samples is in a leaf. Whenever training, we do 5-fold cross-validation on the training data and select the values for both parameters (number of features considered, minimum node size) that give maximum mean performance.

Deep Learning: In keeping with the spirit of the deep learning times, we train a CNN to map directly from pixels to matrix rank. We refer to this as a Scratch CNN in the experiments since it is learned from scratch. Our experiments use a small amount of data, so we adapt a network designed for the MNIST dataset [17] that appears in the examples for [18]. Starting with all images resized to 60×60, our network has architecture $C(5,20) \rightarrow P(4,3) \rightarrow C(5,50) \rightarrow P(4,3) \rightarrow C(4,500) \rightarrow R \rightarrow$ softmax, where $C(k,n)$ denotes a convolutional layer with n filters of size $k \times k$, $P(k,s)$ is max-pooling over a $k \times k$ region with stride s, and R is a rectified linear unit. Empirically, we found that the more aggressive max-pooling than usual helped the network generalize to matrices of other dimensions.

3.2 Implementation Details

We used Piotr Dollar's toolbox [19], Vedaldi et al.'s VLFeat [20], MatConvNet [18], and LIBSVM [21]. *SIFT:* We extract and quantize SIFT on both the gray image and each of the R, G, and B channels separately; each codebook has 256 entries and one codebook is generated on training data per channel. The codebooks are learned once on the 10×10 training set. *Scratch CNN:* We use a learning rate of 10^{-3} in a standard gradient-descent+momentum approach and 1M iterations; to prevent overfitting, we use the first iteration to have validation error within 1% of the final validation error.

4 Experiments

We now rigorously evaluate our approaches for visually guesstimating rank-related matrix properties. *Every figure and table in this section represents a true experiment and actual results. We do not mess around.*

4.1 Dataset

We perform our experiments on a dataset of 10×10 matrices, with 2000 examples of each rank $1, \ldots, 10$, which we split evenly into train and test. When doing binary rank-deficiency classification, we balance class distributions by downsampling the rank deficient class. We generate these matrices by first sampling a matrix $\mathbf{M}$ with entries uniformly and independently sampled from the interval $[0,1]$. We then compute its SVD $\mathbf{M} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ and set $\tilde{\boldsymbol{\Sigma}}$ to $\boldsymbol{\Sigma}$ but with the $r+1, \ldots, n$th entries to 0 and compute $\mathbf{U}\tilde{\boldsymbol{\Sigma}}\mathbf{V}^T$.

We convert each matrix to a 100×100 image, which we store as a PNG. This is done in `MATLAB` by calling the underlying colormapping functionality used by `imagesc` and then upsampling with nearest neighbor. In this paper, we primarily use the traditional and often criticized `jet` colormap, but we also experiment with two linear colormaps, `copper` and `bone`. All colormaps have 255 possible values and are scaled by the min and max of the matrices (i.e., the default of `imagesc`, where no absolute scale is imposed). Note that the many matrices of a variety of ranks may map to the same colormap visualization due to both colormap and PNG quantization.

4.2 Experiments – Features

In this section, we ask the question: what visual features are best suited for visually identifying the rank of a matrix? Is color a useful cue? It was argued in [22] that off-the-shelf pre-trained CNN features are an astoundingly effective baseline for any generic vision task – does this include profoundly unnatural

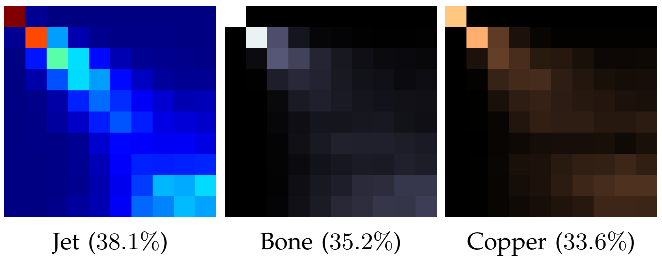


Fig. 3. Confusion matrices and accuracies for different colormaps on the 10-way rank classification problem using dense SIFT and LBP. Note that while there is variation, performance is decidedly above chance across colormaps.

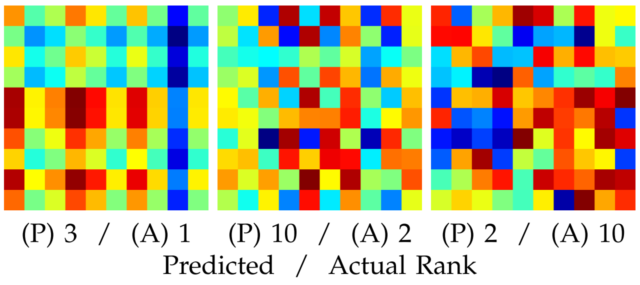


Fig. 5. Failure cases: some deceptive matrices with their (P)redicted and (A)ctual ranks, selected from the most confident mistakes of a RF classifier using dense SIFT and LBP features.

images such as color-mapped matrices? Does learning a specialized CNN work on this task?

We present results in Table 2 showing the performance of various features. In the hand-engineered category, grayscale SIFT seems to perform on par with LBP; adding color considerably improves performance; and using all features does the best. The pretrained CNN does well despite the giant domain shift with both layers do slightly better than grayscale SIFT. However, in keeping with current results in computer vision, training a CNN from scratch consistently does the best. *Note, however, that all feature and method combinations operate at significantly above chance-level.*

4.3 Experiments – Colormaps

One natural question is whether the colormapping scheme affects the visual discrimination between matrices of different ranks. The `jet` colormap (e.g., Figs. 1, 5) in particular has received a lot of criticism for being difficult to interpret in practice by humans. Linear colormaps (i.e., smoothly varying from one color to another) in theory make for easier interpretation by humans. We see whether this holds true for computers as well. We compare the `jet` colormap (named for its origin in astrophysical fluid jet simulation) with `copper` and `bone`. The etymology of `copper` is – we hope for the reader's sake – obvious; `bone` is so named because it looks somewhat like an X-ray and is popular because it lets researchers like us try to pretend to be brain-surgeons.

We run our learning method on matrices with a variety of colormaps and report 10-way classification results in Fig. 3 and 4. Generally, `jet` does the best. The only representation on which it does appreciably worse is the pre-trained CNN; we hypothesize this is because the linear colormaps produce more natural images, whereas the jet colormap's outputs look like noise. Using grayscale SIFT, the results are roughly comparable, which is somewhat surprising as jet is known to convert poorly to grayscale. Nonetheless, while these differences exist, one consistent pattern is that the proposed method works surprisingly well across all colormaps.

4.4 Experiments – Cross Domain

One recent pressing concern in the computer vision community is the biased nature of datasets: models learned on one dataset might not perform even reasonably on another, as reported in [23]. In our case, one might wonder whether a model learned to predict the rank of a 10×10 matrix (with a fixed set of ranks $1, \ldots, 10$) can generalize to matrices of different sizes (e.g., 30×30). To answer this, we train a 10-class random forest on square matrices of dimensions 10, 15, and 30, and test them on different sizes; bag-of-word features are generated using the 10×10 matrix codebooks. These new matrix images have dimensions 150×150 and 300×300 respectively to maintain scale for the SIFT features. CNNs require a fixed input, and so we cannot apply this scaling trick to them.

We train a model to predict ranks $1, \ldots, 10$ for all matrix sizes involved and report results in Table 3. Our method does surprisingly well, performing at around $2.5\times$ chance-level when training on 10×10 matrices and testing on 30×30 matrices and vice-versa. The scratch CNN generalizes well, with the exception of the 30×30 scratch CNN on 10×10 data, which operates at chance level. This is poor generalization as opposed to a bad model to start with: the same model gets 49.7% when testing on 30×30 and 14.6% on 15×15. We believe generalization could be improved by developing an architecture that would enable the row-to-pixel ratio to be constant.

5 Discussion

The success of such a simple approach raises a number of questions, but our method also enables answer to some of these. For instance, we can see what makes a matrix smell rank deficient by analyzing the learned relationship between ϕ and f. We now discuss a few of these questions as well as extensions.

5.1 What does an archetypical rank-k matrix look like and which matrices are tricky to classify?

We can answer each of these questions by looking at the classifier scores; by looking at the most confident

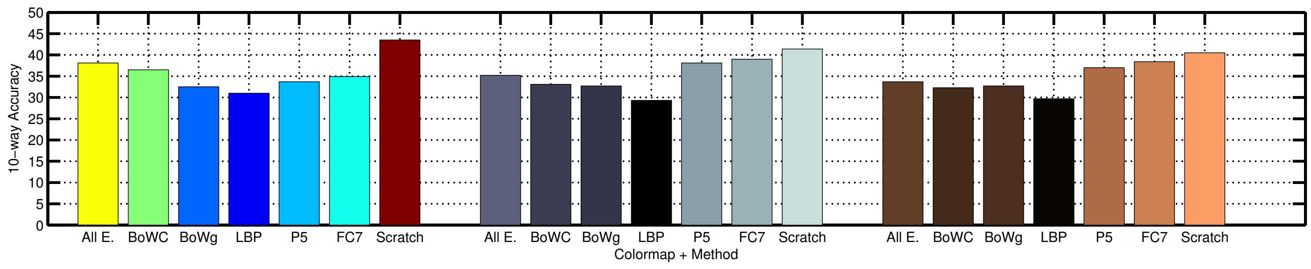


Fig. 4. What colormap is best for predicting matrix rank? (Left to right: jet, bone, copper). While Jet has been criticized widely compared to linear colormaps, it produces the best results with color sift and the from-scratch CNN

TABLE 3
Cross-domain performance: We report the accuracy of the methods on 10-way classification. Although chance on this task is 10%, most of our methods perform substantially better than chance.

Train Dim.	Test Dim.	Random Forest+Engineered *All Eng.*	*Gray SIFT*	*Color SIFT*	*LBP*	RF + Pretrained *pool$_5$*	*fc$_7$*	Scratch CNN *Raw Pixels*
	10×10	38.1%	32.5%	36.5%	31.0%	33.7%	34.9%	**43.5%**
10×10	15×15	33.7%	31.5%	33.5%	27.3%	25.7%	26.1%	**37.9%**
	30×30	**25.9%**	19.1%	25.3%	19.4%	17.7%	17.5%	24.1%
15×15	10×10	33.0%	28.1%	32.0%	26.5%	13.9%	14.4%	**34.1%**
30×30	10×10	**25.8%**	22.7%	23.8%	21.6%	11.5%	11.8%	10.0%

mistakes of the classifier, we can find the most rank-deceptive matrices. We present some archetypical matrices in Fig. 2 according to RF classifier using all features. While the rank 1 matrix archetype is understandable, ranks 2 and up seem inscrutable. Nonetheless, the model is perfectly confident in its assessment of these matrices and is correct a surprising amount of time. Fig. 5 shows the model's confident mistakes. On the left, for instance, is shown a rank-1 matrix with not too much apparent inter-row/column similarity that was mistakenly predicted to rank 3 by the RF.

5.2 What parts of matrices tell us rank?

Given our bag-of-words model, we can answer this by figuring out which codewords help the most in predicting rank as well as their sign (i.e., which codes are most associated with rank-1 rather than rank-10). We solve both by learning a L_2-regularized logistic regression model to predict Rank-i or Rank-j, which we solve with LIBLINEAR [24]. The regularization parameter λ is selected via 5-fold cross-validation to give best average performance. The coefficients of the model $\mathbf{w}$ in terms of magnitude and sign indicate which codewords are indicative of rank deficiency.

We can visualize the informative regions of matrices by replacing pixels with the weight vector of their associated codewords. We show a few examples of this for low rank and high rank matrices using *grayscale* SIFT in Fig. 6. For rank 1, the regions associated with low rank have low frequency, and the codewords associated with high rank occur mainly at the sharp transition from the penultimate and blue column to the last column. The other ranks are a bit harder to interpret, although the one can note the most strongly low-rank regions correspond to flat regions.

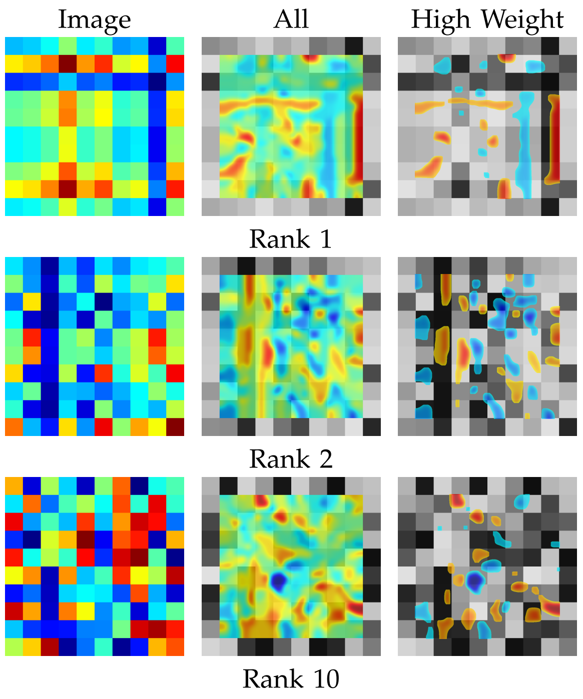


Fig. 6. A visualization of what makes a matrix look rank-deficient according to gray-scale SIFT. We train a logistic regressor to predict rank-k vs. full rank and plot weight-vector coefficients onto the image wherever the codeword appears. Blue is low rank, red high.

5.3 Can We Solve Structured Tasks?

So far, our approach of doing mathematics by learning has only been applied to classification problems. In-

Fig. 7. Examples from our deep visual multiplication net.

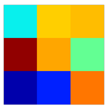
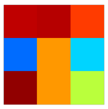
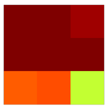
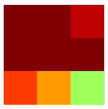
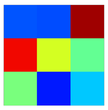
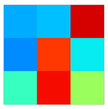
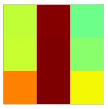
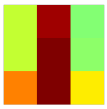
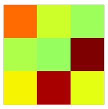
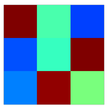
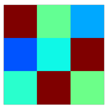
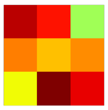
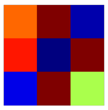
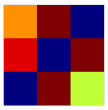
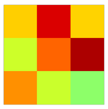
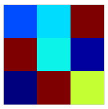
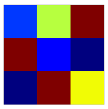
A	B	Pred. $A \cdot B$	True $A \cdot B$	MSE
				0.033
				0.014
				0.012

Fig. 8. Examples from our deep visual matrix inverse net.

A	Predicted A^{-1}	True A^{-1}	MSE
			0.25
			0.49
			1.56

spired by our success in visual rank estimation, we are currently exploring the application of our framework to structured outputs, such as matrices.

In particular, we consider matrix multiplication and inversion, where for centuries mathematicians have relied on hand-crafted, shallow methods. Again, keeping with the times we propose to replace these methods with visual deep learning. To this end we designed a deep learning architecture, differing only in the input for each task: for the multiplication task we used two $M \times M$ concatenated multiplicands as input, whereas in the inversion task there is a single $M \times M$ input. The input is connected to two fully connected layers using ReLU nonlinearities of 512 hidden units each, followed by a fully connected $M \times M$ output layer with no nonlinearity. Dropout regularization was used in all layers. We generate $5 \cdot 10^5$ training examples for each task. In both cases we use 3×3 matrices with each entry independently sampled from a uniform$(0, 1)$ distribution as input, and their "true"[2] product and inverse as outputs.

We then use this data to train the network with stochastic gradient descent on a mean square error (MSE) loss for 100 epochs. Some qualitative predictions on unseen data are shown in Figures 7 and 8. We found the multiplication task to be easily solved by our network architecture, but the inversion task proved much more challenging, as shown by the higher MSE values. We note that this is analogous to humans taking Linear Algebra 101.

5.4 Can This Work On Real Data?

One advantage of our method is that it does not require access to the matrices themselves; but what if we only have a picture of the colormapped matrix? As a proof of concept, we took a cell-phone picture of each part of Fig. 1 of this paper, as shown in Fig. 9, left. We cropped out the matrix from the cell phone picture, as shown in Fig. 9, right. We then resized it and sent it through our scratch CNN. The rank 1 matrix was classified correctly, but both the rank 3 and rank 10 matrix were classified as rank 10. We note, however, that all images have perspective distortion that the CNN did not see at training time.

2. At least according to our matrix library.

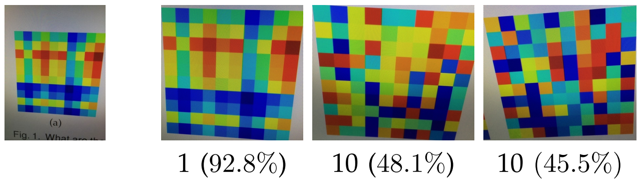


Fig. 9. Results on cropped images from cell-phone pictures of a computer monitor. (Left) sample pre-cropped image; (right) cropped images, their predicted rank, and posterior from the scratch CNN.

5.5 Does the matrix rank network generalize?

In our experiments, we confirmed the reports of [22] that one can use neuron activations from a network pretrained for classification as a strong feature for a variety of tasks; but can we use do the reverse? In other words, can we use activations in our scratch CNN as a feature for image classification?

To evaluate this, we tested our method on the Caltech 101 dataset [25]. We used the concatenated features from the last and second-to-last layers of the rank network (i.e., the softmax responses and the half-wave rectified feature map immediately before) as a feature representation. We then trained a multiclass SVMs (1v1, linear kernel) on top of these representations. We report results in Table 4; results are averaged over $1K$ random samplings of train sets; for test, we use an equal number per-class. While far from state-of-the-art, the numbers are respectable given that the underlying feature representation was trained to estimate matrix ranks.

6 Conclusions

In this paper, we introduced a new problem – visual rank estimation – and demonstrated that it is feasible

TABLE 4
Results on Caltech 101, training a linear SVM over responses from our scratch rank CNN. Chance on this dataset is $\approx 1\%$.

# Samples	5	10	15	20	25
Accuracy	12.6%	17.7%	20.9%	23.3%	25.1%

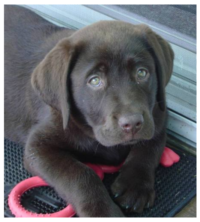
using conventional image classification approaches. Our approach is simple and obtains alarmingly high performance. More importantly, our features also conveys understanding by showing us why some matrices just look low rank and what matrices have surprising rank. We have additionally demonstrated future directions in the form of structured prediction and have demonstrated that our rank predictor CNN can serve as a generic image feature.

References

[1] E. Bareiss, "Sylvester's identity and multistep integer-preserving Gaussian elimination," *Mathematics of Computation*, vol. 22, no. 102, 1968.
[2] J. Bunch and J. Hopcroft, "Triangular factorization and inversion by fast matrix multiplication," *Mathematics of Computation*, vol. 28, no. 125, 1974.
[3] J. Gibson, *The Ecological Approach to Visual Perception*. 1979.
[4] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. of the ACM*, vol. 24, June 1981.
[5] J. Besag, "On the statistical analysis of dirty pictures," *Journal of the Royal Statistical Society, Series B (Methodological)*, vol. 48, no. 3, pp. 259–302, 1986.
[6] J. J. Hopfield and D. W. Tank, ""Neural" computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141–152, 1985.
[7] M. Kennedy and L. Chua, "Neural networks for nonlinear programming," *Circuits and Systems, IEEE Transactions on*, vol. 35, pp. 554–562, May 1988.
[8] A. Cichocki and R. Unbehauen, "Neural networks for solving systems of linear equations and related problems," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 39, pp. 124–138, Feb 1992.
[9] D. F. Fouhey and D. Maturana, "The Kardashian Kernel," in *SIGBOVIK*, 2012.
[10] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *ECCV*, 1994.
[11] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *TPAMI*, vol. 24, no. 7, 2002.
[12] T. Joachims, "Optimizing search engines using clickthrough data," in *KDD*, 2002.
[13] D. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," 2014.
[16] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[18] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," *CoRR*, vol. abs/1412.4564, 2014.
[19] P. Dollár, "Piotr's Image and Video Matlab Toolbox (PMT)." `http://vision.ucsd.edu/ pdollar/toolbox/doc/index.html`.
[20] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms." `http://www.vlfeat.org/`, 2008.
[21] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/ cjlin/libsvm`.
[22] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.
[23] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *CVPR*, 2011.
[24] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
[25] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

David F. Fouhey David F. Fouhey received an A.B. from Middlebury College in Moose Watching in 2011. He is currently a Ph.D. student at the Robotics Institute at Carnegie Mellon University. He likes long walks, Edward Hopper, macchiatos, Jaffa Cakes, and, above all, kvetching. In his copious spare time, he sends fake announcements to the *New York Times'* Wedding Section. He and his colleagues were awarded the *People's Democratic Choice Award* at SIGBOVIK 2013.

Daniel Maturana Daniel comes from Chile. Daniel likes Eat'n Park, bicycling, recycling, and Pabst Blue Ribbon. You won't believe the one weird trick that credit card companies hate that Daniel uses to make money at home thanks to Obama lowering 10-year mortgage rates! He and his colleagues were awarded the *People's Democratic Choice Award* at SIGBOVIK 2013.



Rufus von Woofles Rufus von Woofles is a good boy, isn't he. Yesh he is. Rufus obtained his DoD (Doggy Obedience Diploma), First Class, from Muddy Paws University. Rufus is currently the PI of CHOCOLATE Lab at Carnegie Mellon University, where he leads research on predicting pizza-delivery-man appearance. Rufus likes wagging his tail and belly rubs. Rufus was awarded the *People's Democratic Choice Award* at SIGBOVIK 2013, but he tore it up.