classification on a dataset $\{(\boldsymbol{a}_i, y_i)\}$. For an application to time-varying networks see Kolar et al. (2010).

4. **Logistic + ℓ_2-fusion (LVF):** This model combines logistic loss with the VF setting.

Table 2. Running times (secs) for SLEP and TRIP for optimizing different versions of fused-lasso with increasing input sizes. Both methods were run to satisfy the same convergence criterion.

MODEL	SLEP			TRIP		
n	10^3	10^4	10^5	10^3	10^4	10^5
FS	0.089	1.43	41.80	**0.02**	**0.10**	**0.86**
VS	0.16	1.26	35.77	**0.02**	**0.10**	**0.90**
LFL	**0.21**	15.01	144.81	0.78	**5.35**	**53.88**
LVF	0.86	**0.02**	132.13	**0.81**	0.15	**11.24**

Synthetic data. We first compare TRIP equipped with our proximity solvers with the approach of Liu et al. (2010). Here random matrices $\boldsymbol{A} \in \mathbb{R}^{n \times m}$ are generated, whose entries are selected to follow a zero mean, unit variance normal distribution. We fix $m = 100$, and set $\lambda_1 = \lambda_2 = 0.01$. Then, we sample matrices with number of columns n varying as 10^3, 10^4, and 10^5. To select the vector of responses $\boldsymbol{y}$, we use the formula $\boldsymbol{y} = \mathrm{sgn}(\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{v})$, where $\boldsymbol{x}_t$, and $\boldsymbol{v}$ are random vectors whose entries have variances 1 and 0.01, respectively. The numerical results are summarized in Table 2, where we compare SLEP (version 4.0) (Liu et al., 2009) against the TRIP-based approach.[3] While for smaller matrices with $n = 10^3$ both methods run similarly fast, as the size of the input matrices increases, the TRIP-based fused-lasso solvers run much faster than SLEP.

Real Data. We tested each of the four FL models on binary classification tasks for the following microarray datasets: ArrayCGH (Stransky et al., 2006), Leukemias (Golub et al., 1999), Colon (U. Alon et al., 1999), Ovarian (Rogers et al., 2005) and Rat (Hua et al., 2009). Each dataset was split into three equal parts (ensuring both classes are present in every split) for training, validation and test. The penalty parameters where found by grid search in the range $\lambda_1, \lambda_2 \in [10^{-3}, 10^1]$ to maximize classification accuracy on the validation splits.

Table 3 shows test accuracies. We see that in general, logistic-loss based FL models yield better classification accuracies than those based on least-squares. This result is natural: logistic-loss is more suited for classification in tasks like the ones proposed for these datasets. Regarding the TV-regularizer, three out of five datasets seem to be insensitive to this choice, though the $\mathrm{Tv}_1^{1\mathrm{D}}$-penalty performs better for Ovarian, while $\mathrm{Tv}_2^{1\mathrm{D}}$ works best for ArrayCGH.

[3]Both TRIP and SLEP are implemented in MATLAB; only the crucial proximity operators are implemented in C.

Table 3. Classification accuracies on microarray data.

DATASET	FL	VF	LFL	LVF
ARRAYCGH	73.6%	**78.9%**	73.6%	73.6%
LEUKEMIAS	92.0%	92.0%	**96.0%**	**96.0%**
COLON	**77.2%**	**77.2%**	**77.2%**	**77.2%**
OVARIAN	**88.8%**	83.3%	77.7%	77.7%
RAT	67.2%	65.5%	**70.4%**	**70.4%**

5.2. Results for 2D-TV

We now show application of our two-dimensional $\mathrm{Tv}_{1,1}^{2\mathrm{D}}$-proximity solver. We are not aware of natural applications for two or higher-dimensional $\mathrm{Tv}_{2,2}^{2\mathrm{D}}$-proximity, so we do not discuss it further. The most basic and natural application of our $\mathrm{Tv}_{1,1}^{2\mathrm{D}}$-proximity is to image denoising. Among the vast number of denoising methods, we compare against the well-established method based on the classic ROF-TV model (Rudin et al., 1992). This model takes an $n \times n$ noisy image $\boldsymbol{Y}$ and denoises it by solving

$$\min_{\boldsymbol{X}} \quad \tfrac{1}{2}\|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}}^2 + \lambda \mathrm{Tv}_{\mathrm{rof}}(\boldsymbol{X}), \tag{21}$$

where the ROF version of TV is defined as

$$\mathrm{Tv}_{\mathrm{rof}}(\boldsymbol{X}) = \sum\nolimits_{1 \le i,j < n} \|(\nabla x)_{i,j}\|_2,$$
$$(\nabla x)_{i,j} = \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ x_{i,j+1} - x_{i,j} \end{bmatrix}.$$

That is, the TV operator is applied on the discrete gradient over the image. This TV regularization is known as isotropic TV, in contrast to our anisotropic TV. Although often the isotropic version $\mathrm{Tv}_{\mathrm{rof}}$ is preferred, for some applications anisotropic TV shows superior denoising. We show a simple example that illustrates this setting naturally, namely, denoising of two-dimensional barcodes (Choksi et al., 2010). We apply our 2D-TV operator to this setting and compare against the isotropic model which we solved using the state-of-the-art PDHG method (Zhu & Chan, 2008). For further reference we also compare against: (i) the anisotropic TV solver proposed in (Friedman et al., 2007); (ii) an adapted (anisotropic) PDHG solver obtained easily by modifying the original formulation; and (iii) a median filter. We note that the stepsize selection rules recommended for PDHG, failed to produce fast runtimes when applied to anisotropic TV. Thus, to make PDHG competitive, we searched for optimal stepsize parameters for it by exhaustive grid search.

Table 4 presents runtimes and Improved Signal-to-Noise Ratio (ISNR) values obtained for a series of denoising experiments on barcode images that were corrupted by additive (variance 0.2) and multiplicative (variance 0.3) gaussian noise. To compensate for the loss of contrast produced by TV filtering, intensity values are rescaled to the range