APERFEIÇOAMENTO DE MODELO DE PREVISÃO DE CONFLITOS EM RODOVIAS ATRAVÉS DE DEEP LEARNING

Felipe Caleffi
Universidade Federal de Santa Maria – UFSM, Campus Cachoeira do Sul - RS
Helena Beatriz Bettella Cybis
Laboratório de Sistema de Transportes – LASTRAN; Universidade Federal do Rio Grande do Sul – UFRGS

RESUMO
Modelos de previsão de risco de colisão em tempo real, que dependem de dados de tráfego, podem ser úteis em sistemas de gerenciamento dinâmico que buscam melhorar a segurança do tráfego. Modelos têm sido propostos para prever a ocorrência de conflitos e o risco de colisão para melhorar proativamente a segurança. Este trabalho apresenta o esforço para aperfeiçoar o modelo de previsão de conflitos proposto em Caleffi *et al.* (2017), através de técnicas de análise multivariada, para prever a probabilidade de conflitos entre veículos em um segmento da rodovia BR-290/RS, situado na região metropolitana de Porto Alegre – RS. Neste trabalho, são empregadas técnicas de *deep learning* (redes neurais), traçando um comparativo entre as capacidades preditivas dos modelos. Resultados indicam que o modelo de redes neurais melhorou a acurácia de previsão de 76,28% para 85,92% - com uma melhora na sensibilidade (classificação correta de conflitos efetivamente ocorridos) de 87,02% para 92,55%.

ABSTRACT
Real-time collision risk prediction models relying on traffic data can be useful in dynamic management systems seeking at improving traffic safety. Models have been proposed to predict conflict occurrence and collision risk in order to proactively improve safety. This paper presents the effort to improve the conflict prediction model proposed in Caleffi *et al.* (2017) by means of multivariate analysis techniques to predict collision probability between vehicles in a segment of the highway BR-290/RS, located in the Porto Alegre - RS metropolitan region. In this work, deep learning techniques (neural networks) are used, drawing a comparison between the models predictive capabilities. Results indicate that the neural network model improved the predictive accuracy from 76.28% to 85.92% - with an improvement in sensitivity (correct classification of conflicts actually occurred) from 87.02% to 92.55%.

1. INTRODUÇÃO

A segurança nas rodovias é um tema muito estudado na engenharia de transportes. Como as colisões são uma medida direta da segurança do tráfego, muitos estudos focalizam em encontrar uma relação entre características operacionais da rodovia e o risco de colisão. Modelos de previsão de colisões em tempo real são frequentemente usados para avaliar o risco de colisão com base em dados agregados de tráfego coletados a partir de detectores instalados no pavimento ou de câmeras de vigilância (Li *et al.*, 2014).

O risco de colisão pode ser estimado como a probabilidade de ocorrências de conflito com base nas características existentes de fluxo de tráfego na via, como velocidade média, fluxo, ocupação ou densidade. Um conflito de tráfego acontece quando dois ou mais usuários se aproximam no espaço e no tempo, de tal forma que há risco de colisão se seus movimentos permanecerem inalterados (Davis *et al.*, 2011). A previsão de conflitos ajuda a identificar condições perigosas de tráfego em que estratégias proativas de prevenção de acidentes são necessárias para mitigar a alta probabilidade de colisão. Diversos estudos propuseram modelos para prever o risco de acidentes em rodovias com base nas características operacionais do tráfego. Uma revisão sistemática sobre os impactos das características de tráfego na ocorrência de acidentes em rodovias é apresentada por Roshandel *et al.* (2015), e resume os principais trabalhos da área.

No Brasil, as condições de tráfego nas rodovias de múltiplas faixas não são homogêneas. O tráfego nestas rodovias apresenta um comportamento altamente heterogêneo devido a vários

fatores: (i) intensidades de fluxo e composições de tráfego diferentes, (ii) limites de velocidade diferentes para classes de veículos leves e pesados, e (iii) agressividade dos condutores, que leva a um elevado número de trocas de faixa e ultrapassagens. Esses fatores geram altos níveis de congestionamento e heterogeneidade do tráfego entre faixas, as ultrapassagens são frequentes e, nestas condições, muito perigosas. Portanto, o desenvolvimento de modelos destinados a prever a ocorrência de colisões ou conflitos tem potencial para ajudar a identificar condições de tráfego adversas, e a propor estratégias para aumentar a segurança.

O modelo publicado em Caleffi *et al.* (2017) empregou técnicas de análise multivariada para modelar a relação entre as características de tráfego e a probabilidade da ocorrência de conflitos na rodovia BR-290/RS. O número reduzido de amostras referentes a acidentes neste trecho de rodovia não permite a construção de um modelo de previsão de acidentes. Assim, esta proposta foca em conflitos de tráfego. A relevância deste estudo tem origem nas particularidades do estudo de caso – com foco em um segmento que representa o comportamento de tráfego típico de uma rodovia brasileira de múltiplas faixas com rampas de acessos com elevado fluxo. Resultados indicaram que o modelo apresentou uma acurácia média de classificação de 76% e classificou corretamente 87% dos conflitos efetivamente ocorridos (avaliados por meio de sensibilidade).

Para este trabalho, o objetivo foi aperfeiçoar o modelo de previsão de conflitos proposto em Caleffi *et al.* (2017) através de técnicas de *deep learning* (redes neurais), traçando um comparativo entre as capacidades preditivas dos modelos.

2. TRECHO EM ESTUDO E CONSTRUÇÃO DO BANCO DE DADOS

O trecho em estudo corresponde ao segmento da rodovia BR-290/RS, na região metropolitana da cidade de Porto Alegre – RS. Este trecho de rodovia possui 3 faixas de tráfego com uma rampa de acesso. Esta seção é o principal acesso à cidade de Porto Alegre. A localização foi selecionada entre outras seções de rodovias devido a vários critérios: a qualidade das câmeras de monitoramento de tráfego, a extensão do congestionamento e a existência de rampa de acesso de alto fluxo que causam distúrbios no fluxo de tráfego. Vídeos de trânsito foram gravados usando câmera de monitoramento distribuída para cobrir esta rampa de acesso, sendo posicionada acima da rodovia para atingir uma altura de visualização adequada para cobrir toda a área do gargalo 50 metros a jusante da rampa de acesso.

A coleta de dados através de vídeo foi realizada em dias de semana sob condições climáticas favoráveis. Um total de 150 horas de dados de tráfego foram registrados em maio de 2013, nos quais 120 horas com boa visibilidade (aproximadamente 60 horas de dados para cada rampa de acesso). Os vídeos gravados foram posteriormente analisados em laboratório para obter dados de fluxo de tráfego, velocidades e conflitos. Um estudante de graduação treinado foi designado para rever todos os vídeos para garantir que critérios consistentes foram aplicados para identificar conflitos. Os dados de conflitos foram registados seguindo os passos propostos em Huang *et al.* (2013). Além disso, a metodologia de coleta de dados foi corroborada por outros membros da pesquisa, para garantir a consistência dos dados. Maiores detalhes quanto a coleta de dados é descrita em Caleffi *et al.* (2017).

A Figura 1 apresenta uma relação fluxo-velocidade para dados coletados através de laços indutivos no km 96, para todo o mês de maio de 2013, que ilustra as diferenças de tráfego entre faixas – particularidades típicas do tráfego nas rodovias brasileiras. A faixa da esquerda (Faixa 1) apresenta velocidades e taxas de fluxo mais altas, enquanto a faixa da direita (Faixa 3)

apresenta as velocidades e taxas de fluxo mais baixas. A Faixa 3 recebe principalmente caminhões e ônibus, justificando taxas de fluxo e velocidades reduzidas. Os dados foram agregados em intervalos de 5 minutos.

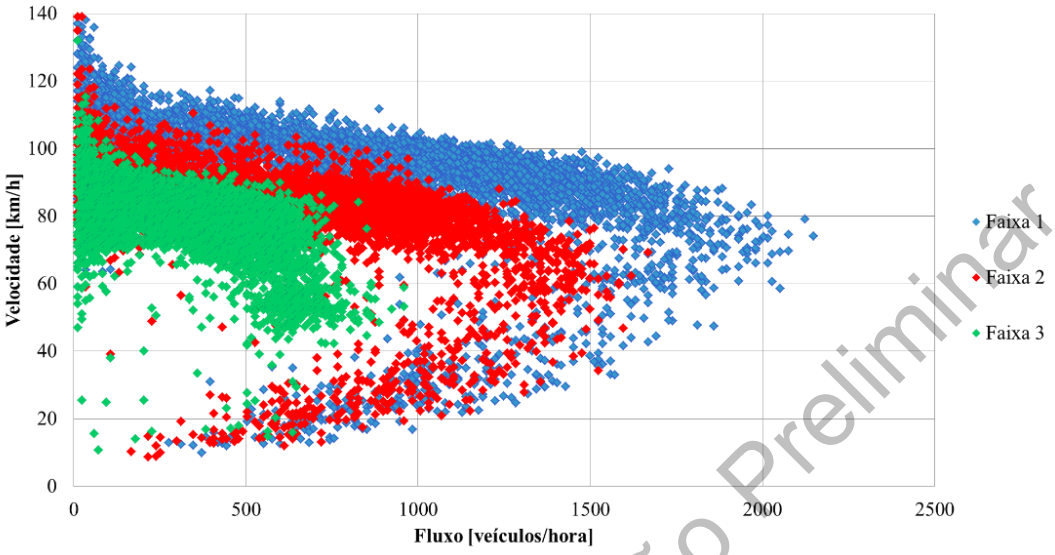

Figura 1: Relação fluxo-velocidade por faixa – km 96.

3. APERFEIÇOAMENTO DO MODELO DE PREVISÃO DE CONFLITOS

O modelo de previsão de conflitos proposto em Caleffi *et al.* (2017) apresentou, primeiramente, o conjunto de variáveis coletadas através de câmeras de monitoramento – que potencialmente ofereciam explicações sobre a ocorrência de conflitos. A seguir, foram propostos dois índices alternativos baseados na distância de Bhattacharyya (*Bhattacharyya Distance – BD*) (Coleman e Andrews, 1979) e Análise de Componentes Principais (*Principal Component Analysis – PCA*) (Rencher, 2002), com o objetivo de identificar as variáveis mais relevantes para categorizar ocorrências em classes de conflito ou não conflito. Esses índices orientaram a remoção de variáveis menos importantes no processo de seleção de variáveis. Depois de cada variável ser excluída, a capacidade de classificação das variáveis restantes foi avaliada para categorizar eventos em conflito ou não. O subconjunto selecionado de variáveis foi então inserido em um modelo de Análise Discriminante Linear (*Linear Discriminant Analysis – LDA*) (Rencher, 2002) desenvolvido para estimar a probabilidade de conflito.

O método proposto foi aplicado a dados que descrevem eventos de tráfego que resultaram em conflito ou não-conflito; Tais dados foram constituídos por 96 observações (48 eventos conflitantes e 48 eventos não conflitantes) e descritas por 36 variáveis. Dado o desempenho superior do modelo PCA, foi inserido as 5 variáveis mais relevantes sugeridas pela estrutura PCA em um modelo LDA adaptado para prever a ocorrência de conflitos na rodovia avaliada em tempo real. A função LDA resultante, denominada Modelo de Previsão de Conflitos (CPM), é representada na equação (1).

$$CPM = -4{,}784 + 0{,}001(Total.Flow_5) + 0{,}00019(Total.Flow_{10}) - 67{,}496(Diff.Std.Dev.Occ_5) + 4{,}173(Diff.Std.Dev.Occ_{10}) + 7{,}339(Coeff.Var.Speed_5) \quad (1)$$

Onde: $Total.Flow_5$ e $Total.Flow_{10}$ são os fluxos totais para os intervalos 0-5min e 5-10min anteriores ao evento de conflito, respectivamente; $Diff.Std.Dev.Occ_5$ e $Diff.Std.Dev.Occ_{10}$ são a diferença entre os desvios padrão da ocupação das faixas para os intervalos 0-5min e 5-10min; e $Coeff.Var.Speed_5$ é o coeficiente de variação das velocidades para o intervalo 0-5min.

O modelo apresenta um Lambda de Wilk = 0.565 e um valor de p = 0.001, indicando que o modelo explica corretamente as diferenças entre as classes e é significativo. Os valores CPM positivos indicam a ocorrência de um evento de conflito, enquanto os valores negativos apontam para um estado sem conflito. Os resultados indicaram que o modelo LDA-PCA superou o LDA alinhado com a distância Bhattacharyya. A abordagem LDA-PCA obteve uma acurácia de classificação média de 76% e classificou corretamente 87% dos conflitos efetivamente ocorridos (avaliados por meio da sensibilidade).

3.1. Deep learning

Técnicas de *deep learning* utilizam algoritmos de aprendizagem de máquina (*machine learning*) para modelar sistemas que requerem um alto nível de abstração. Algoritmos de *deep learning* utilizam múltiplas camadas de processamento, têm sido usados em diversos campos e tem produzido alguns dos melhores resultados dentre as possíveis soluções com *machine learning* (Deng e Yu, 2013).

Em um modelo de *deep learning*, um vetor de entrada é frequentemente mapeado em um vetor de saída por meio de um conjunto de funções não lineares. Vetores de entrada podem ser, no caso de acidentes: as características do acidente, como comportamento do motorista, e características da rodovia, veículo e ambiente. O vetor de saída são as classes correspondentes do acidente. O *deep learning* permite que modelos computacionais aprendam representações hierárquicas de dados com múltiplos níveis de abstração em diferentes camadas de processamento (Abdelwahab e Abdel-Aty, 2001).

A vantagem das redes neurais de *deep learning* sobre as técnicas estatísticas é que elas envolvem um procedimento de mapeamento mais geral, ou seja, uma função específica não é necessária na construção de modelos. No entanto, essas técnicas podem ser tratadas como métodos de caixa preta, se a arquitetura de rede não for cuidadosamente projetada e seus parâmetros não forem otimizados (Abdelwahab e Abdel-Aty, 2001).

Vários estudos investigaram Redes Neurais (RN) e métodos de *deep learning* em aplicações relacionadas a transporte (Alkheder *et al.*, 2017; Duan *et al.*, 2016; Lv *et al.*, 2015; Yu *et al.*, 2017). Por exemplo, Abdelwahab e Abdel-Aty (2001) empregaram RN para prever a gravidade do dano ao motorista a partir de vários fatores de um acidente (ou seja, características do motorista, do veículo, da pista e do ambiente). Em seu estudo, o modelo de RN desenvolvido foi melhor que um modelo probit ordenado. Delen *et al.* (2006) aplicaram RN para modelar a gravidade da lesão de acidentes em uma rodovia usando 17 parâmetros significativos. O modelo RN foi utilizado para prever os níveis de gravidade da lesão.

Alkheder *et al.* (2017) usaram métodos de RN para prever a gravidade da lesão (leve, moderada, grave, morte) de acidentes de trânsito em Abu Dhabi. A precisão geral do modelo para os dados de treinamento e testes foi de 81,6% e 74,6%, respectivamente. Zeng e Huang (2014) propuseram um algoritmo de treinamento e um método de otimização de estrutura de rede para previsão de severidade de acidentes com colisão. Seus resultados indicaram que o algoritmo de treinamento proposto apresentou melhor desempenho que o algoritmo tradicional de retro-

propagação. A RN otimizada, que continha menos nós do que a RN totalmente conectada, alcançou uma precisão razoável de previsão. Além disso, os modelos de RN totalmente conectadas e otimizadas superaram um modelo logit ordenado.

De modo geral, *deep learning* permite uma composição e um projeto de estruturas de rede flexíveis com módulos de multicamadas e, portanto, espera-se que o desempenho dos modelos seja melhor do que os modelos tradicionais de RN.

3.2. Modelo de redes neurais (RN)

Os modelos de redes neurais (RN) são mecanismos de processamento de informações inspirados nos sistemas nervosos biológicos. Os neurônios podem ser organizados de diversas formas, mas em geral, são organizados em camadas. Normalmente existe uma camada de entrada, em que os dados entram na rede, existe uma ou mais camadas ocultas, em que os dados são processados e uma camada de saída, onde os dados são manipulados para ficarem no seu formato de saída (Russell, 1996).

Matematicamente, uma RN é uma função complexa, projetada para aprender com os dados coletados. Dependendo de seus mecanismos de aprendizagem, as RNs geralmente podem ser divididos em dois grupos, ou seja, supervisionados e não supervisionados. As RNs supervisionadas são adequadas apenas para o aprendizado de amostras com rótulos (aqueles com resultados esperados, como observações de colisões). As RNs supervisionadas clássicas incluem redes *perceptron* de várias camadas (*multilayer perceptron* – MLP) e função de base radial (RBF). Pelo contrário, as RNs não supervisionadas são geralmente usadas para aprender com amostras não rotuladas (aquelas sem saídas esperadas). O MLP, conhecido como um aproximador universal, é a RN supervisionada mais popular para mineração de dados. A Figura 2 mostra uma estrutura MLP desenvolvida com neurônios totalmente conectados (Haykin, 2009).

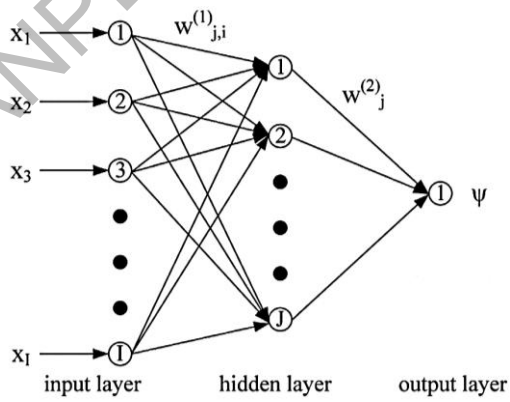

Figura 2: Estrutura MLP com neurônios totalmente conectados (Haykin, 2009).

O processo de aprendizado de todas as RNs pode ser com dados durante a execução, de forma online, ou com dados captados anteriormente. A técnica mais comum de treinamento é chamada de *back-propagation*. Durante esse tipo de treinamento, para cada dado de entrada é calculado o erro entre a saída e o valor esperado real. Após o cálculo do erro, o sistema propaga esse erro para as camadas anteriores, ajustando os pesos entre as camadas. Seguindo esse processo, novos parâmetros são definidos para as entradas dos neurônios segundo uma taxa de aprendizado

arbitrária, que pode ser tanto estática, quanto dinâmica, de forma a aproximar a saída do valor real esperado (Russell, 1996).

Repetindo esses passos para diferentes entradas no sistema, os pesos se ajustam de forma a tentar encontrar o menor erro de saída. Para uma base de dados extensa o erro tende a aproximar de um valore mínimo, local ou absoluto, para aquele banco de dados usado no treinamento. Caso o erro não chegue num valor mínimo, treina-se a rede múltiplas vezes em repetições chamadas de *epochs* (iterações).

Considere um banco de dados contendo *N* atributos. Cada atributo é representado por um nó x_i ($i = 2, \cdots I$) na camada de entrada. Além disso, um nó de entrada, com $x_1 = 1$, é adicionado. O número de unidades *I* na camada de entrada é:

$$I = N_1 + \sum_{n=1}^{N_2} (m_n - 1) + 1 \tag{2}$$

Para resolver problemas de engenharia, um modelo MLP com 3 camadas (uma camada oculta) é suficiente, pois estudos apontam que RNs com apenas uma camada oculta são menos propensos a ficar preso em um mínimo local durante o treinamento da rede (de Villiers e Barnard, 1993).

3.3. Treinamento da rede

Para ajustar bem os dados de treinamento, o número de neurônios na camada oculta deve ser suficientemente grande. Se for assumido como *J*, então o peso de conexão entre o nó oculto, $j = 1, \cdots, J$ e o nó de entrada *i*, $i = 1, \cdots, I$ é $w_{j,i}^{(1)}$. A função hiperbólica, tanh(·), que é uma função de transferência de sigmoide ímpar, é usada para todos os nós ocultos.

Na camada de saída, a única unidade, ψ, representa a frequência de conflitos esperada. $w_j^{(2)}$ denota o peso da conexão entre o nó de saída e o nó oculto *j*, $j = 1, \cdots, J$. Uma função linear é empregada como a função de transferência para o nó de saída. Como resultado, temos:

$$\psi = \sum_{j=1}^{J} w_j^{(2)} \tanh\left(\sum_{i=1}^{I} w_{j,i}^{(1)} x_i\right) \tag{3}$$

O método proposto por Haykin (2009), que pode acelerar a convergência do processo de treinamento da rede, é adotado neste estudo. Este método utiliza o algoritmo BP (*back-propagation*) para modificar os pesos de conexão da rede de acordo com o erro de cálculo de um exemplo no conjunto de dados de treinamento a cada vez. Durante o treinamento, o valor de erro é realimentado pelas saídas da rede para ajustar os pesos em cada camada. O treinamento não será interrompido até que o erro atinja o valor limite ou a rede neural obtenha o tempo máximo de treinamento (Wang *et al.*, 2011). Os passos do algoritmo BP proposto são os seguintes:

1. Inicialização. $w_{j,i}^{(1)}$ (j = 2, · · ·, J; i = 1; · · ·, I) e $w_{j,i}^{(2)}$ (k = 1, · · ·, K; j = 1; · · ·, J) são selecionados aleatoriamente a partir de duas distribuições uniformes diferentes.

As médias de ambas as distribuições são iguais a zero e suas variâncias são 1/J e 1/K, respectivamente.

2. Construção da *epoch*. Aleatoriamente organizar os dados de treinamento em uma *epoch* de 1 para *M*. Para cada padrão (referente à "observação" em modelos estatísticos) *m* na *epoch*, conduza os cálculos na etapa 3–5.

3. Cálculo de avanço. Calcula as saídas dos nós na camada oculta e na camada de saída, $H_j(m)$, $\psi_k(m)$, e os erros de cálculo $e_k(m)$ através:

$$v_j^{(1)}(m) = \sum_{i=1}^{I} w_{j,i}^{(1)} \cdot x_i(m), H_j(m) = g_j(v_j^{(1)}(m)) \quad (4)$$

$$v_k^{(2)}(m) = \sum_{j=1}^{J} w_{k,j}^{(2)} \cdot H_j(m), \Psi_k(m) = g_k(v_k^{(2)}(m)) \quad (5)$$

$$e_k(m) = o_k - \Psi_k(m) \quad (6)$$

Onde $g_j(v)$ e $g_k(v)$ são as funções de ativação dos neurônios. A função hiperbólica, tanh(·), que é uma função de ativação sigmoide ímpar, é usada para todos os neurônios da rede.

4. Cálculo para trás. Calcule os gradientes locais $\delta_k^{(2)}(m)$ e $\delta_j^{(1)}(m)$ dos neurônios da camada de saída e da camada oculta e os valores de correção $\Delta w_{k,j}^{(2)}(m)$ e $\Delta w_{j,i}^{(1)}(m)$ de seus pesos de conexão através:

$$\delta_k^{(2)}(m) = e_k(m) \cdot g'_k(v_k^{(2)}(m)) \quad (7)$$

$$\Delta w_{k,j}^{(2)}(m) = a(m) \cdot \Delta w_{k,j}^{(2)}(m-1) + \eta(m) \cdot \delta_k^{(2)}(m)\Psi_k(m) \quad (8)$$

$$\delta_j^{(1)}(m) = g'_j(v_j^{(1)}(m)) \sum_{k=1}^{K} \delta_k^{(2)}(m) w_{k,j}^{(2)}(m) \quad (9)$$

Onde $a(m)$ e $\eta(m)$ são o momento e o tamanho do passo, respectivamente. Ambos diminuem com m: $\eta(m) = ns/n_s + m\ \eta(0)$, $a(m) = ns/n_s + m\ a(0)$, em que n_s é um parâmetro que controla a velocidade decrescente.

5. Atualizando. Atualize todos os pesos de conexão na rede:

$$w_{j,i}^{(1)} = w_{j,i}^{(1)} + \Delta w_{j,i}^{(1)}(m), w_{k,j}^{(2)} + \Delta w_{k,j}^{(2)}(m) \quad (10)$$

6. Verificando critérios de convergência. No final de uma *epoch*, se o critério de convergência for atendido, o treinamento da rede termina; caso contrário, volte ao passo 2.

Além do algoritmo BP, o treinamento da rede também utilizou o algoritmo Stochastic Gradient Descent (SGD) em Tensorflow numa CPU pessoal (Core i5 com 8 GB RAM). O algoritmo SGD usa alguns exemplos do vetor de treinamento de entrada e calcula as saídas e os erros e, em seguida, ajusta os pesos. O processo é recorrente para muitos conjuntos pequenos de exemplos até que a média da função objetivo pare. O método SGD pode determinar um bom conjunto de pesos quando comparado com outras técnicas de otimização (Bottou e Bousquet, 2008; Wilson *et al.*, 2017). O treinamento foi executado em um tamanho de lote 32 com 100 *epochs*. O melhor modelo foi obtido usando SGD com o decaimento de (*decay* = 0,90) e um momento de (*momentum* = 0,90). A melhor taxa de aprendizado foi de 0,01.

3.4. Mitigação do overfitting

Redes com funções mais complexas podem ter um desempenho de generalização melhor à medida que aprendem diferentes representações em cada uma das suas camadas. No entanto, redes complexas podem facilmente causar *over-fit* (excesso de ajuste) nos dados de treinamento, produzindo baixa capacidade de generalização e desempenho de testes. *Overfitting* (ou sobrecarregamento) ocorre quando um modelo de rede com alta capacidade ajusta o ruído nos dados em vez do relacionamento subjacente.

Portanto, para evitar o ajuste excessivo, três técnicas padrão foram utilizadas. Estas foram a injeção de ruído gaussiano nos dados de treinamento (Zur *et al.*, 2009), usando uma função de ativação ReLU nas camadas ocultas (LeCun *et al.*, 2015), e subsequentemente aplicando a técnica de *dropout* (Hinton *et al.*, 2012). O *dropout* leva a grandes melhorias no desempenho de previsão do modelo. Quando uma técnica de *dropout* é aplicada, a capacidade de generalização do modelo é aprimorada porque a rede é forçada a aprender várias representações independentes dos dados. Uma probabilidade de 30% foi usada na técnica de *dropout*. Isso porque baixa probabilidade tem efeito mínimo e uma alta probabilidade resulta em sub-aprendizado pela rede.

3.5. Ajuste de hiper-parâmetros

Os hiper-parâmetros do modelo RN foram selecionados através de uma pesquisa sistemática de grade implementada em scikit-learn (Pedregosa *et al.*, 2011) usando 100 *epochs*. Embora a pesquisa sistemática de grade exija alto custo computacional, melhores resultados podem ser obtidos ajustando-se sistematicamente os valores dos hiper-parâmetros. Os parâmetros do modelo com a maior acurácia são apresentados na Tabela 1.

Tabela 1: Parâmetros otimizados usados no modelo.

Hiper-parâmetro	Melhor Valor	Descrição
Tamanho de lote	32	Número de casos de treinamento sobre os quais a atualização do SGD é calculada.
Função de perda	*Categorical crossentropy*	A função objetivo ou função de pontuação de otimização também chamada de *logloss multiclass*, apropriada para alvos categóricos.
Otimizador	SGD	Otimizador de descida de gradiente estocástico.
Taxa de aprendizado	0,01	Taxa de aprendizado usada pelo otimizador do SGD.
Gradient momentum	0,90	Momento de gradiente usado pelo otimizador do SGD.
Decay	0,90	Taxa de decaimento de aprendizado sobre cada atualização.

4. RESULTADOS EXPERIMENTAIS E DISCUSSÃO

O modelo RN proposto foi implementado em Python usando a estrutura de *deep learning* TensorFlow de código aberto desenvolvida pelo Google (Abadi *et al.*, 2016). O TensorFlow possui recursos de diferenciação automática e compartilhamento de parâmetros, o que permite que uma ampla gama de arquiteturas seja facilmente definida e executada. A rede proposta foi treinada com 158 mostras e validada com 40 mostras usando o framework Tensorflow. O algoritmo de otimização SGD foi aplicado, com um tamanho de lote de 32 e uma taxa de aprendizado de 0,01, como destacado na Tabela 1.

A Figura 3 mostra o desempenho da acurácia do modelo RN calculado para 100 *epochs* (iterações) usando os conjuntos de dados de treinamento (80%) e de validação (20%). Em geral, a precisão do modelo nos conjuntos de dados de treinamento e validação aumenta após cada iteração com flutuações. As flutuações na precisão são devido ao uso da técnica de *dropout* no modelo, que resulta em um treinamento diferente durante cada iteração. A técnica de *dropout*, usada para evitar o ajuste excessivo, introduz algumas das aleatoriedades na rede.

Na primeira iteração, a precisão foi de 58,32% e 64,88% para dados de treinamento e testes, respectivamente. À medida que o modelo treina durante a primeira passagem pelos dados, as perdas por treinamento e validação diminuem, indicando que o modelo está aprendendo a estrutura dos dados de conflitos de tráfego, e, possivelmente, suas correlações temporais. Nas primeiras e consecutivas iterações, a perda na validação não aumentou significativamente e foi sempre menor que a perda no treinamento, indicando que a rede não sobrecarregou os dados de treinamento (não houve *overffiting*) e generalizou com precisão os dados de validação não vistos. Após 100 *epochs*, a precisão de validação do modelo foi de 85,92%. A sensibilidade do modelo (fração de eventos conflitantes corretamente classificados como conflito) foi de 92,55%. A especificidade do modelo (fração de eventos não-conflitantes corretamente classificados como não-conflito) foi de 76,43%.

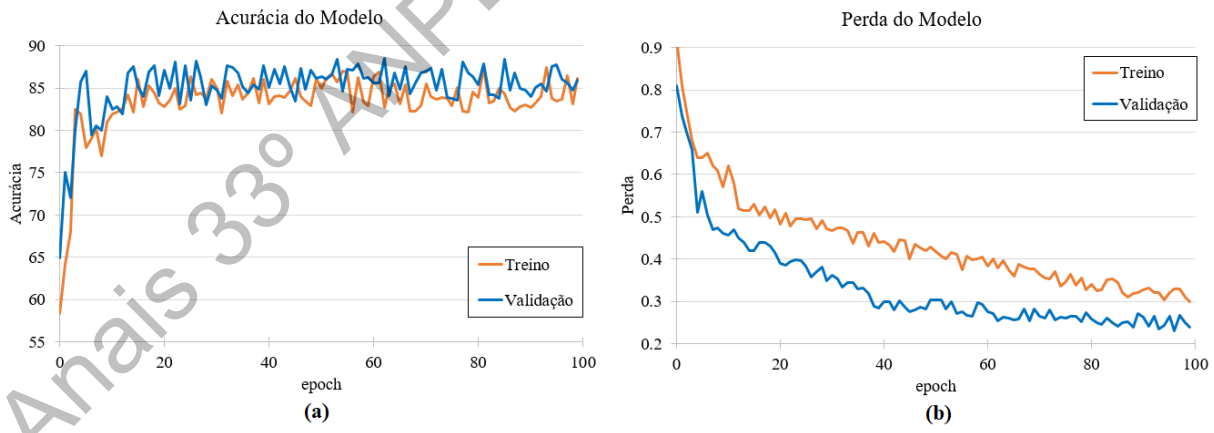

Figura 3: Acurácia (a) e perda (b) do modelo RN calculado para 100 *epochs*.

Geralmente, em um modelo de *deep learning*, vários módulos e multicamadas podem ser empilhados uns sobre os outros, de modo que seja significativo analisar a profundidade da rede (número de camadas ocultas) para entender o comportamento da rede. A Tabela 2 mostra a precisão do modelo RN com vários números de camadas densas usadas (*Dense layers*). A melhor precisão foi obtida usando duas camadas densas (totalmente conectadas) com 64 unidades ocultas (*hidden units*). Esse modelo obteve acurácias de treinamento e validação de 86,11% e 85,92%, respectivamente. Quando outra camada densa foi adicionada à rede, as

precisões de treinamento e validação foram reduzidas. O modelo começou a ter *overffiting* quando mais de cinco camadas densas foram usadas.

Tabela 2: Acurácia do treinamento e validação do modelo RN com números diferentes de camadas densas.

Número de Camadas Densas	Acurácia do Treinamento (%)	Acurácia da Validação (%)
2	86,11	85,92
3	85,14	85,21
4	82,55	84,87
5	80,35	84,06
6	78,42	64,56

Além disso, o efeito do número de camadas MLP na precisão do modelo foi avaliado. A Tabela 3 mostra que a melhor precisão de treinamento (86,11%) e validação (85,92%) pode ser alcançada com uma rede de uma camada MLP.

Tabela 3: Acurácia do treinamento e validação do modelo RN com números diferentes de camadas MLP.

Número de Camadas LSTM	Acurácia do Treinamento (%)	Acurácia da Validação (%)
1	86,11	85,92
2	85,33	85,51
3	81,78	81,96

O tempo médio de treinamento e teste por iteração com tamanho de lote de 32 do modelo RN também foi medido. O modelo gasta em média cerca de 137 milissegundos por iteração durante o treinamento e apenas cerca de 11 milissegundos por previsão para novos exemplos não vistos. Embora este experimento mostre a eficiência computacional do modelo, também é importante notar que o tempo de treinamento pode ser aumentado reduzindo o tamanho do lote ou o comprimento da sequência, e também aumentando o volume dos dados de treinamento.

4.1. Comparação com o modelo de Caleffi *et al.* (2017)

Nesta seção são apresentadas as comparações do modelo RN, com o modelo desenvolvido em meu doutorado, e apresentado em Caleffi *et al.* (2017).

Tabela 4: Desempenho de classificação no conjunto de teste para ambos os índices.

Desempenho da Classificação Média no Conjunto de Testes	Modelo RN (%)	LDA – PCA (%)
Acurácia	85,92	76,28
Sensibilidade	92,55	87,02
Especificidade	76,43	68,11
Desvio Padrão da Acurácia	9,75	9,22
Desvio Padrão da Sensibilidade	13,12	12,50
Desvio Padrão da Especificidade	13,64	13,42

Esta comparação evidencia que o modelo RN foi capaz de melhorar a acurácia do modelo em cerca de 10%, e a sensibilidade em 5,5%, demonstrando que a técnica de *deep learning* possui significativo potencial de otimização na classificação correta de conflitos de tráfego.

5. CONSIDERAÇÕES FINAIS

Neste artigo, um modelo de Rede Neural (RN) foi desenvolvido para prever a probabilidade de colisão entre veículos em um segmento da rodovia BR-290/RS, situado na região metropolitana de Porto Alegre – RS. Uma arquitetura de rede otimizada foi determinada através de uma pesquisa sistemática de grade para os hiper-parâmetros de rede sub-ótimos. Vários hiper-parâmetros do modelo foram críticos para alcançar a maior precisão de validação. O algoritmo de otimização usado foi o SGD com taxa de aprendizado, momento e decaimento do peso de 0,01, 0,90 e 0,90, respectivamente. Além disso, a técnica de *dropout* ajudou a reduzir a complexidade do modelo e também a chance de ocorrer *overffiting* no conjunto de dados de treinamento. O modelo RN obteve a melhor acurácia de validação de 85,92%, com uma sensibilidade do modelo de 92,55%.

Apesar de o modelo RN ter aperfeiçoado o modelo previamente apresentado em (Caleffi *et al.*, 2017), estudos futuros devem se concentrar no desenvolvimento de estruturas de rede mais flexíveis e otimizadas, com o intuito de aumentar ainda mais a acurácia. Futuros estudos teóricos são encorajados a focar na melhoria da capacidade do modelo RN em representar variáveis e estruturas de dados, e para armazenar dados em longos períodos de tempo.

Agradecimentos

O presente trabalho foi realizado com apoio do CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil. Bolsista do CNPq – Brasil: Felipe Caleffi.

REFERÊNCIAS BIBLIOGRÁFICAS

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., e Devin, M. (2016) Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.

Abdelwahab, H. T., e Abdel-Aty, M. A. (2001) Development of Artificial Neural Network Models to Predict Driver Injury Severity in Traffic Accidents at Signalized Intersections. *Transportation Research Record: Journal of the Transportation Research Board*, *1746*(1), 6–13. doi:10.3141/1746-02

Alkheder, S., Taamneh, M., e Taamneh, S. (2017) Severity Prediction of Traffic Accident Using an Artificial Neural Network. *Journal of Forecasting*, *36*(1), 100–108. doi:10.1002/for.2425

Bottou, L., e Bousquet, O. (2008) The Tradeoffs of Large Scale Machine Learning. *Advances in Neural Information Processing Systems*, *20*, 161–168.

Caleffi, F., Anzanello, M. J., e Cybis, H. B. B. (2017) A multivariate-based conflict prediction model for a Brazilian freeway. *Accident Analysis & Prevention*, *98*, 295–302. doi:10.1016/j.aap.2016.10.025

Coleman, G. B., e Andrews, H. C. (1979) Image segmentation by clustering. *Proceedings of the IEEE*, *67*(5), 773–785. doi:10.1109/PROC.1979.11327

Davis, G. A., Hourdos, J., Xiong, H., e Chatterjee, I. (2011) Outline for a causal model of traffic conflicts and crashes. *Accident; analysis and prevention*, *43*(6), 1907–19. doi:10.1016/j.aap.2011.05.001

de Villiers, J., e Barnard, E. (1993) Backpropagation neural nets with one and two hidden layers. *in IEE Transactions on Neural Networks*, *4*(1), 136–141. doi:DOI: 10.1109/72.182704

Delen, D., Sharda, R., e Bessonov, M. (2006) Identifying significant predictors of injury severity in traffic accidents using a series of artificial neural networks. *Accident Analysis and Prevention*, *38*(3), 434–444. doi:10.1016/j.aap.2005.06.024

Deng, L., e Yu, D. (2013) Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing*, *7*(2013), 197–387. doi: 10.1561/2000000039

Duan, Y., Lv, Y., Liu, L., Wang, Y. (2016) An efficient realization of deep learning for traffic data imputation. *Transportation Research Part C: Emerging Technologies*, *72*, 168–181. doi:10.1016/j.trc.2016.09.015

Haykin, S. (2009) *Neural network and learning machines*. (Third Edit.). Hall., Pearson Prentice, Canada.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., e Salakhutdinov, R. R. (2012) Improving neural

networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 1–18.arxiv.org/abs/1207.0580

Huang, F., Liu, P., Yu, H., e Wang, W. (2013) Identifying if VISSIM simulation model and SSAM provide reasonable estimates for field measured traffic conflicts at signalized intersections. *Accident; analysis and prevention*, *50*, 1014–24. doi:10.1016/j.aap.2012.08.018

LeCun, Y., Bengio, Y., Hinton, G.(2015) Deep learning. *Nature Methods*, *521*, 436-444. doi10.1038/nmeth.3707

Li, Z., Ahn, S., Chung, K., Ragland, D. R., Wang, W., e Yu, J. W. (2014) Surrogate safety measure for evaluating rear-end collision risk related to kinematic waves near freeway recurrent bottlenecks. *Accident; analysis and prevention*, *64*, 52–61. doi:10.1016/j.aap.2013.11.003

Lv, Y., Duan, Y., Kang, W., Li, Z., e Wang, F. Y. (2015) Traffic Flow Prediction with Big Data: A Deep Learning Approach. *IEEE Transactions on Intelligent Transportation Systems*, *16*(2), 865–873. doi:10.1109/TITS.2014.2345663

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vandeplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., e Duchesnay, É. (2011) Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. Obtido de http://scikit-learn.sourceforge.net.

Rencher, A. C. (2002) Methods of Multivariate analysis. *Wiley Interscience: New York, USA*.

Roshandel, S., Zheng, Z., e Washington, S. (2015) Impact of real-time traffic characteristics on freeway crash occurrence: Systematic review and meta-analysis. *Accident; analysis and prevention*, *79*(August), 198–211. doi:10.1016/j.aap.2015.03.013

Russell, I. (1996) Neural Networks. *The UMAP Journal*, *14*(1). doi:10.1016/S0893-6080(13)00287-6

Wang, J. Z., Wang, J. J., Zhang, Z. G., e Guo, S. P. (2011) Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, *38*(11), 14346–14355. doi:10.1016/j.eswa.2011.04.222

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., e Recht, B. (2017) The Marginal Value of Adaptive Gradient Methods in Machine Learning. *Advances in Neural Information Processing Systems 30*.

Yu, R., Li, Y., Shahabi, C., Demiryurek, U., e Liu, Y. (2017) Deep Learning: A Generic Approach for Extreme Condition Traffic Forecasting. *Proceedings of the 2017 SIAM International Conference on Data Mining*, 777–785. doi:10.1137/1.9781611974973.87

Zeng, Q., e Huang, H. (2014) A stable and optimized neural network model for crash injury severity prediction. *Accident Analysis and Prevention*, *73*, 351–358. doi:10.1016/j.aap.2014.09.006

Zur, R. M., Jiang, Y., Pesce, L. L., e Drukker, K. (2009) Noise injection for training artificial neural networks: A comparison with weight decay and early stopping. *Medical Physics*, *36*(10), 4810–4818. doi:10.1118/1.3213517