INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

DEPARTMENT OF ELECTRONICS, SYSTEMS, AND INFORMATICS.

MASTER IN COMPUTING SYSTEMS.

Implementation and development of Traffic Speed and Flow prediction through Artificial Neural Networks.

Submitted by: Ulises Salvador Vega Arteaga

To obtain the degree of

MASTER IN COMPUTING SYSTEMS

Advisors:
Dr. Raul Campos Rodríguez
Dr. Riemann Ruiz Cruz.

Tlaquepaque, Jalisco, México; June 10th, 2016.

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN SISTEMAS COMPUTACIONALES

DESARROLLO DE UNA RED NEURONAL ARTIFICIAL PARA LA PREDICCIÓN DE VELOCIDAD Y FLUJO DEL TRÁFICO.

Trabajo recepcional que para obtener el grado de

MAESTRO EN SISTEMAS COMPUTACIONALES

Presenta: Ulises Salvador Vega Arteaga

Directores:
Dr. Raul Campos Rodríguez
Dr. Riemann Ruiz Cruz

Tlaquepaque, Jalisco, México; Junio de 2016.

ACKNOWLEDGEMENTS

I want to thank Alexander Klotz, and Yao Zhai from Continental for the proposal of the idea of this work that I hope is of relevance to new technologies in automated (self-driving) cars.

I want to thank ITESO, CONACYT (No. 502171), and Continental Guadalajara Services S.A. de C.V. for its support with the scolarship provided to obtain this degree.

I want to thank also to my advisors Dr. Riemann Ruiz Cruz, and Dr. Raul Campos Rodríguez for their continuous support and effort during the realization of this work.

AGRADECIMIENTOS

Quiero agradecer a Alexander Klotz, y Yao Zhai de Continental por la propuesta de la idea de este trabajo, esperando que los resultados de esta sean de utilidad para las nuevas tecnologías de automóviles autónomos.

Quiero agradecer al ITESO, y a CONACYT (como becario No. 502171), Continental Guadalajara Services S.A. de C.V., por la beca que me proporcionaron para la realización de estos estudios.

Quiero agradecer a los doctores Raul Campos Rodríguez, y Riemann Ruiz por su continuo esfuerzo y soporte durante la realización de este trabajo.

DEDICATION.

I want to dedicate this thesis to my parents Pedro Vega Muñoz and Elizabeth Arteaga Ortiz, as well as to my brother Arquímedes Uriel Vega Arteaga for being a supportive family.

DEDICATORIA

El autor dedica esta tesis a mis padres Pedro Vega Muñoz, y Elizabeth Arteaga Ortiz, así como también a mi hermano Arquímedes Uriel Vega Arteaga por ser mi apoyo diario.

SUMMARY

In this work we introduce the most recent techniques to predict traffic flow and speed. This work is composed of the following sections: an introduction, a state of the art, and conclusions sections.

In the introduction section we see the importance of being able to predict the traffic conditions for speed and flow; we set our hypothesis that is that using the Levenberg-Marquardt training algorithm we'll be able to find a global minimum for the problem of predicting traffic conditions; we also specify our general objective that is developing efficient algorithms for the traffic prediction for its variables speed and flow; as well as establishing that our scientifical novelty is using the Levenberg-Marquardt as Artificial Neural Network training algorithm.

In the state of the art section we present the summarized contents of the most outstanding research papers about traffic prediction. We start this section by reviewing the concept of Big Data because the information obtained for the Department of Transportation from California is a text file, and this is considered as unstructured data; additionally, there is a huge volume of information available for the problem of study, and it also arrives at a constant speed of 5 minutes every measurement. We also introduce some information related to autonomous cars which are going to be on the roads from 2020 and ahead. This information is obtained from the National Highway Traffic Safety Administration of the United States of America.

We continue with the presentation of an approach that uses two statistical algorithms for traffic prediction. This information cover spatial impact, and accidents, and construction events. Additionally, it compares the results of outstanding research done with Artificial Neural Networks, and Statistical Methods, to their own statistical method that consists of two statistical methods embedded into only one by using a threshold used to determine which statistical method should be used and when, depending on the road conditions.

We also present the results of traffic speed prediction by data mining techniques and a comparative to Artificial Neural Networks. Additionally, we introduce a section that analyze the problem of traffic prediction but only with Artificial Neural Networks done by the company SIEMENS in Germany. As we can see many studies use Artificial Neural Networks and that's because they are very tolerant to noise sensitivity to which Statistical Methods are not. For example: noise in traffic prediction arises from a malfunctioning sensor, or a measurement record lost.

We then introduce the Los Angeles Department of Transportation (LA DOT) infrastructure used to obtain the speed and flow measurements as well as the set of programs develop by the author of this thesis to

retrieve the information from LA DOT, to process this information and calculate the prediction of flow and speed for a specific sensor.

We continue with the problem of traffic prediction. In the process of predicting traffic flow, and speed we made our first approach using a Nonlinear Autoregressive Neural Network with External Input in MATLAB and we obtained promising results. However, this Artificial Neural Network does not have the ability to predict multiple outputs, and we transformed it to a Feedforward Neural Network also from MATLAB. The results obtained are impressive because they reduce dramatically the traffic prediction errors. In order to validate our results with the ones in the international research community we use the data from 95 days which is equivalent to 3 months, which is the commonly reported amount of time studied in this kind of problems. We also present an optimization process for the Feedforward Neural Network for both problems speed and flow prediction. Finally, we present a numerical sensibility analysis in order to determine how robust is our Artificial Neural Network.

To close this thesis we present our conclusions as a success of using the Levenberg-Marquardt algorithm to train an Artificial Neural Network for the problem of traffic prediction, and we set up the possibilities of exploring the recently found result by using the learning techniques of deep learning.

RESUMEN

En este trabajo se presentan las técnicas más recientes de predicción de flujo y velocidad de tráfico. Este trabajo se compone de las siguientes secciones: introducción, estado del arte, y conclusiones.

En la introducción observamos la importancia de predecir el flujo y velocidad de tráfico y establecemos la hipótesis de que usando el algoritmo de Levenberg-Marquardt para entrenar una Red Neuronal Artificial (RNA) podemos alcanzar un mínimo global en este problema, es decir, un mínimo de error. Además, establecemos como nuestro objetivo principal el desarrollar algoritmos eficientes para el problema de predicción de tráfico entendiendo por eficientes que tengan un mínimo de error. Además, declaramos que nuestra novedad científica es utilizar el algoritmo de Levenberg-Marquardt para entrenar una RNA.

En la sección del estado del arte presentamos una síntesis de los resultados más importantes en cuanto a investigación referente a la predicción de flujo y velocidad de tráfico. Comenzamos esta sección con un repaso sobre "*Big Data*" porque la información obtenida del Departamento de Transporte de California (DOT CA) es un archivo de texto, mismo que se considera como información no estructurada. Adicionalmente, existe una gran cantidad de información y cada archivo de texto llega cada 5 minutos con información sobre todo el condado de Los Ángeles California. También presentamos información relacionada con los carros autónomos que estarán en circulación a partir del 2020.

Continuamos la sección del estado del arte con la presentación de un estudio que implica el uso de dos algoritmos estadísticos como si fuera uno sólo, mediante el uso de umbral que nos ayuda a decidir cuando usar uno u otro algoritmo dependiendo de las condiciones del tráfico.

También presentamos un reporte de investigación basado en minería de datos para predecir la velocidad del tráfico comparado con el uso de RNAs. Proseguimos, con una presentación de una prueba de concepto realizada por la empresa SIEMENS de Alemania, para predecir el flujo y velocidad de tráfico. Como podemos observar las RNAs, se utilizan demasiado y esto es porque son muy robustas ante el ruido. El ruido en el problema de la predicción del tráfico se origina por un sensor que no funciona y cuya lectura se pierde o por la perdida de un registro -archivo de texto- en el proceso de captura de información.

Enseguida procedemos con una introducción de la infraestructura del departamento de transporte de Los Ángeles California misma de la cual se obtuvieron las medidas de flujo y velocidad del tráfico. Se presenta la arquitectura de los programas utilizados para el intercambio y procesamiento de la información y así calcular las predicciones de las variables ya mencionadas.

Para hacer la predicción de las variables de flujo y velocidad describimos nuestra primera aproximación usando una RNA tipo NARX del programa MATLAB obteniendo resultados prometedores. Sin embargo, esta red tipo NARX no tiene posibilidad de predecir multiples salidas por lo que la transformamos a una red tipo *“Feedforward”* de MATLAB. Los resultados obtenidos son impresionantes pues el error de predicción es reducido drásticamente. Con la finalidad de validar nuestros resultados con la comunidad internacional se colectó información de un periodo de 95 días que es el estándar utilizado en la comunidad investigadora. También presentamos un proceso de optimización de las RNAs con el fin de mejorar el desempeño de la RNA. Finalmente, cerramos la sesión de estado del arte con un análisis de sensibilidad de nuestra RNA con la finalidad de saber que tan robusta es nuestra RNA.

Para cerrar el presente trabajo presentamos nuestras conclusiones mismas que mencionan el éxito obtenido al utilizar el algoritmo de Levenberg-Marquardt como método de entrenamiento de una RNA. Apuntalamos nuestro trabajo haciendo referencia a posibles futuros trabajos que se pueden realizar utilizando técnicas como el aprendizaje profundo o *“Deep learning”* o el uso de Hadoop para el procesamiento de la información como *“Big Data”*.

TABLE OF CONTENTS.

FIGURES INDEX.

TABLE INDEX

ACRONYMS.

ANN	Artificial Neural Network.
ARIMA	Autoregressive Integrated Moving Average
CALTRANS	California Department of Transportation
CHP	California Highway Patrol
DB	Database
ES	Exponential Smoothing
ESC	Electronic Stability Control
FHT	Foothill Transit
FTP	File Transfer Protocol
HAM	Historical Average Model
H-ARIMA	Hybrid ARIMA
HLDI	Highway Loss Data Institute
ITESO	Instituto Tecnológico y de Estudios Superiores de Occidente
ITS	Intelligent Transportation Systems.
LA Metro	Los Angeles Metro
LADOT	Los Angeles Deparment of Transportation
LBT	Long Beach Transit
MAPE	Mean Absolute Percent Error
MATLAB	Mathematical Software called MATLAB
MLP	Multi Layer Perceptron
MPP	Massive Parallel Processing
MSE	Mean Square Error
MySQL	Name of an OpenSource Database
NARX	Nonlinear Autoregressive Neural Network with External Input
NHTSA	National Highway Traffic Safety Administration

NNET	Neural Network
PBS	Pattern Based Strategy
PeMS	California Performance Measurment System
PoC	Proof of Concept
RFID	Radio Frequency Identification
RMSE	Root Mean Square Error
RS	Rough Set Theory
SMP	Symmetric Multiprocessing
TFS	Traffic Forecast Server
TMC	Traffic Monitor Client
TMS	Traffic Management Server
VMZ Berlin	Traffic Management Central for the city of Berlin

1. INTRODUCTION

1.1. Background.

The traffic flow prediction is a problem that arises from continuous arrival of cars, trucks, and other type of vehicle to crossing points in road networks. They arrive at different times and also at different locations that are interconnected one to each other. This creates a spatio-temporal problem because a congestion in one place will influence another place traffic flow.

In this way every driver has a traffic flow problem as well as the transportation authorities.

In this document will explore the most recent developments of algorithms for traffic prediction also known as traffic forecasting, identify its advantages and disadvantages.

1.2. Justification

We want to be able to predict traffic flow because this reduces gas consumption, vehicle accidents, optimizes roads usage and reduces time invested in travels i.e: from home to work and viceversa.

1.3. Problem.

The problem relies in current prediction methods that are too sensitive to noise and to sudden changes in traffic flow. These impediments block the driver from having effective planning for her travels and fall inadvertently in car congestions.

1.4. Hypothesis

Given the good behavior to obtain models that predict time series by artificial neural networks, we propose the usage of complex artificial neural networks to predict the traffic flow of a city.

1.5. Objectives

1.5.1. General objective:

The objective is to develop efficient algorithms that analyze and extract useful information from structured/unstructured data so that the traffic flow speed can be predicted.

As a start we have established the following two objectives and if time allows something else could be added:

1.5.2. Specific objectives:

1. The algorithm should use articificial neural networks, exploring training algorithms other than the backpropagation for example: the Levenberg-Marquardt.

2. At least the traffic should be predicted half an hour ahead, in steps of 5 minutes.

1.6. Scientific, or technological novelty, or contribution.

First, the specific contribution of this work is the experimental demonstration that to predict traffic flow and speed we can use an ANN using Levenberg-Marquardt algorithm. Second, there is no need to use two algorithms topredict traffic efficiently.

2. STATE OF THE ART.

2.1. What is big data?.

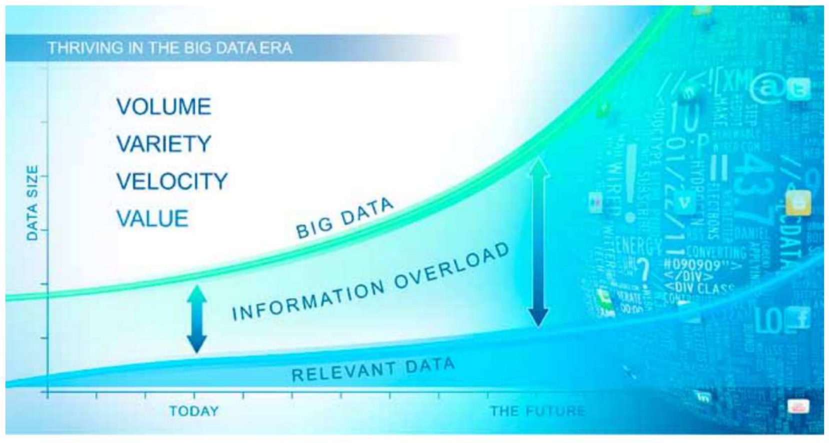
In 2001, Doug Laney defined Big Data with the three V's[6]:

- **Volume:** The volume is affected with storage through the years, unstructured data from several social media information sources, sensor and machine to machine data being collected. Storing large amounts of data creates the question of which is the relevant information to be stored.
- **Velocity:** reacting in almost real-time fashion for many data coming from sensors, RFID tags, smart metering, has become a reality for many applications on many fields.
- **Variety:** Data comes in different types: structured and unstructured. Examples of structured are numerical traditional databases. Examples of unstructured are: text documents, email, video, audio, stock ticker data, and financial transactions. Managing these types of data is a great challenge for companies around the world.

At SAS (www.sas.com), they consider another two V's:

- **Variability:** The data flows can be very inconsistent with periodic peaks. If something is trending in social media, it can be daily, seasonal, or event-triggered peak data loads can be challenging to manage for structured and unstructured data.
- **Complexity:** There are multiple sources of information. And it takes great effort link, match, cleanse, and transform data across systems. However, it is a need to connect and correlate relationships, hierarchies, and multiple data linkages or your data can quickly spiral out of control.

A key point in handling big data is the ability to understand big data to transform the business. The old rules of data management are found to apply no more when it comes to big data[7].



Determining relevant data is key to delivering value from massive amounts of data.

Figure 1.- Graphical depict of the V's in big data.

Some technical approaches to solve big data analytics are: grid computing, in-database processing, and in-memory analytics. SAS has introduced the concept of an analytical data

warehouse that surfaces for analysis only the relevant data from the enterprise data warehouse, for simpler and faster processing. Price of storage has decreased from $16 dollars in 2000 to less than $0.07 today. Computing models such as parallel processing, clustering, virtualization, grid environments and cloud computing, coupled with high-speed connectivity, have redefined what is possible. Three key factors in extracting value from big data are [7] :
Information management for big data. Manage data as a strategic, core asset, with ongoing process control for big data analytics.
High-performance analytics for big data. Gain rapid insights from big data and the ability to solve increasingly complex problems using more data.
Flexible deployment options for big data. Choose between on-premises or hosted, cloud computing.

Information Management for Big Data [7].
Many organizations already struggle to manage their existing data. Big data will only add complexity to the issue. What data should be stored, and how long should we keep it? What data should be included in analytical processing, and how do we properly prepare it for analysis? What is the proper mix of traditional and emerging technologies? Big data will also intensify the need for data quality and governance, for embedding analytics into operational systems, and for issues of security, privacy and regulatory compliance. Everything that was problematic before will just grow larger.

Analysing blogs, emails, product catalogs, and wiki articles to extract important concepts from these information to look at the links among them to identify and assign weights to millions of terms and concepts, so that we can look at this context to assess data as it streams into the organization. This analysis identifies the relevant data that should be pushed to the enterprise datawarehouse or to high performance analytics.

High performance analytics for big data[7].

Successful organizations can't wait days or weeks to look at what's next. Decisions need to be made in minutes or hours, not days or weeks.

High-performance analytics also makes it possible to analyze all available data (not just a subset of it) to get precise answers for hard-to-solve problems and uncover new growth opportunities and manage unknown risks – all while using IT resources more effectively.

Accelerated processing of huge data sets is made possible by four primary technologies:
• Grid computing. A centrally managed grid infrastructure provides dynamic workload balancing, high availability and parallel processing for data management, analytics and reporting. Multiple applications and users can share a grid environment for efficient use of hardware capacity and faster performance, while IT can incrementally add resources as needed.
• In-database processing. Moving relevant data management, analytics and reporting tasks to where the data resides improves speed to insight, reduces data movement and promotes better data governance. Using the scalable architecture offered by third-party databases, in-database processing reduces the time needed to prepare data and build, deploy and update analytical models.

• In-memory analytics. Quickly solve complex problems using big data and sophisticated analytics in an unfettered manner. Use concurrent, in-memory, multiuse access to data and rapidly run new scenarios or complex analytical computations. Instantly explore and visualize data. Quickly create and deploy analytical models. Solve dedicated, industry-specific business challenges by processing detailed data in-memory within a distributed environment, rather than on a disk.

Flexible deployment options for big data.

High performance analytics can be deployed on the cloud, a dedicated high performance appliance, or in the existing on-premises infrastructure, whichever best suits your needs. Using symmetric multiprocessing (SMP) to massive parallel processing (MPP) running on tens, hundreds, or even thousands of servers, a flexible architecture enables organizations to take advantage of hardware advances and different processing options, while extending the value of original investments.

Ideas about big data for vehicles connectivity.

Capitalizing in the connected vehicle. Connecting data of the vehicle with data of the environment where it is-such as weather, traffic and other vehicles or fleets-. To send drivers real time alerts about impeding external conditions.[1] Examples of it are: during a traffic congestion the ability to propose alternate routes for more fluid transit in a programmed or current route, proposing alternate routes to floods or inundations directly on a screen or on a mobile smart phone. Another example might be using driving information history that starts from the last turn on of the car suggesting speeds and driving patterns that can help economize fuel consumption. Another example might be internal heating or cooling conditions/patterns in accordance to external sensed temperature and weather conditions i.e.: rain, snow, heat, etc.

2.2. Crash avoidance technologies[3].

Many new vehicles offer advanced crash avoidance features. The systems started out as options on a few luxury models and have steadily spread to more of the fleet. Advanced technologies assist the driver with warnings or automatic braking to help avoid or mitigate a crash. These include front crash prevention, lane departure warning, blind spot detection, adaptive headlights and park assist and backover prevention. Advances also are being made in intelligent transportation systems that allow vehicles to communicate with one another or with road infrastructure to help avoid crashes.
Front crash prevention and adaptive headlights are reducing insurance claims. HLDI (Highway Loss Data Institute) found fewer claims under property damage liability coverage, which pays for damage to vehicles that an at-fault driver hits, for models with forward collision warning with automatic braking than for the same vehicles that weren’t equipped with the technology. Adaptive headlights also are reducing property damage liability claims. So far, HLDI hasn’t been able to quantify the real-world effects of other advanced crash avoidance systems.

Electronic stability control is an older — and proven — crash avoidance feature. Standard on 2012 and later models, ESC is an extension of antilock brake technology that helps drivers maintain control of their vehicles on curves and slippery roads. ESC lowers the risk of a fatal single-vehicle crash by about half and the risk of a fatal rollover by as much as 80 percent.

National Highway Traffic Safety Administration (NHTSA) defines vehicle automation as having five levels[4]:
No-Automation (Level 0): The driver is in complete and sole control of the primary vehicle controls – brake, steering, throttle, and motive power – at all times.
Function-specific Automation (Level 1): Automation at this level involves one or more specific control functions. Examples include electronic stability control or pre-charged brakes, where the vehicle automatically assists with braking to enable the driver to regain control of the vehicle or stop faster than possible by acting alone.
Combined Function Automation (Level 2): This level involves automation of at least two primary control functions designed to work in unison to relieve the driver of control of those functions. An example of combined functions enabling a Level 2 system is adaptive cruise control in combination with lane centering.
Limited Self-Driving Automation (Level 3): Vehicles at this level of automation enable the driver to cede full control of all safety-critical functions under certain traffic or environmental conditions and in those conditions to rely heavily on the vehicle to monitor for changes in those conditions requiring transition back to driver control. The driver is expected to be available for occasional control, but with sufficiently comfortable transition time. The Google car is an example of limited self-driving automation.
Full Self-Driving Automation (Level 4): The vehicle is designed to perform all safety-critical driving functions and monitor roadway conditions for an entire trip. Such a design anticipates that the driver will provide destination or navigation input, but is not expected to be available for control at any time during the trip. This includes both occupied and unoccupied vehicles.

2.3. Utilizing Real-World Transportation Data for Accurate Traffic Prediction.

In [8] it is stated that they achieve improvements in traffic patterns predictions increase by 67% and 78% in short-term (5 minutes) and long-term (30 minutes). If an accident gets involved improvements are done in a 91%. This improvements are done over autoregression, neural nets, and smoothing techniques taking information from databases “such as such as Caltrans, City of Los Angeles Department of Transportation (LADOT), California Highway Patrol (CHP), Long Beach Transit (LBT), Foothill Transit (FHT) and LA Metro”. Currently there are lots of sensors deployed on the road that can be of help to determine traffic situation in real time in a road. They were able to identify:

- temporal patterns during rush hours, or

- spatial impacts of accidents

An algorithm such as ARIMA cannot capture sudden changes at the temporal boundaries of rush hours. This can be improved by incorporating historical data to improve predictions of the road by 30 minutes. However and improved ARIMA cannot predict accidents and its spatial impact. “Again, historical data can be used to identify similar accidents, i.e., with similar severity, similar location and during the similar time,”. .. “for generic time-series, the observations made in the immediate past are usually a good indication of the short-term future. However, for traffic time-series, this is not true at the edges of the rush hours, due to sudden speed changes.

1. *Effect of Prediction Horizon (h):* Historical Average Model (HAM) is much better at predicting the traffic flow when h >6 (30 minutes) than normal ARIMA for the same time. See figure 2 for details.

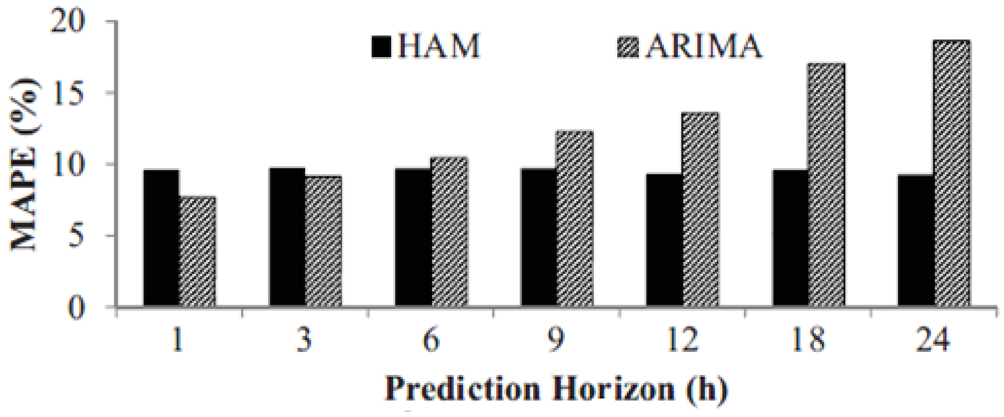


Figure 2.- Behavior of ARIMA and HAM algorithms in different prediction horizons.

Effect of Rush Hour Boundaries: As we can observe in figure 3 HAM beats ARIMA in prediction. ARIMA takes a small delay to take into account the sudden change in speed.

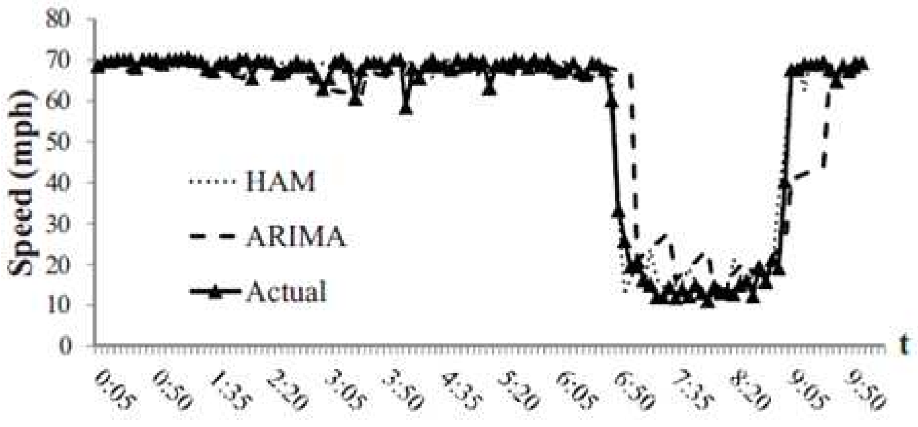


Figure 3.- Effect of rush hours in HAM, and ARIMA.

The authors in [8] proposed an hybrid forecasting model. Depending on ratio of error

$\lambda = Err_{ARIMA} / (Err_{ARIMA} + Err_{HAM})$

if the ratio is less than 0.5 they use ARIMA to predict the traffic otherwise they use HAM. The effect of rush hours in the algorithms can be seen on figure 4. Where we can see that there are three pikes at the boundary of rush hours meaning that HAM outperforms ARIMA.

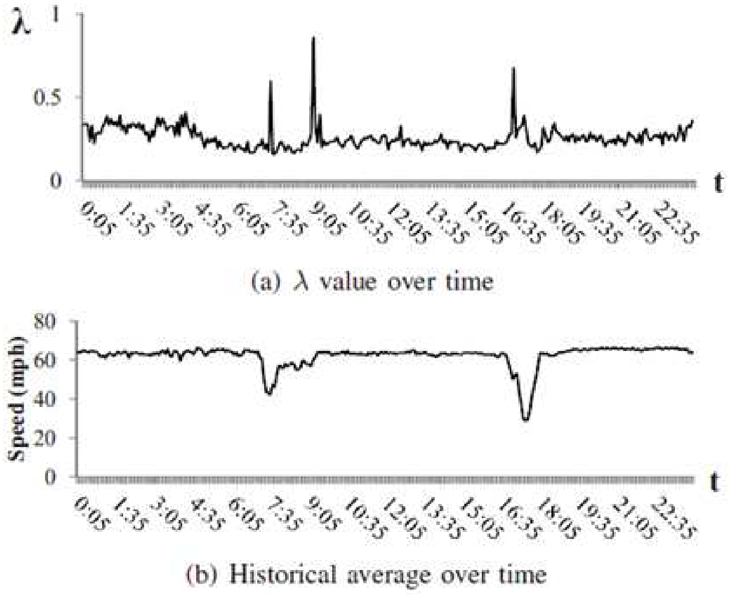


Figure 4 .- Effect of rush hours over λ

Event impact analysis:

“HAM can hardly react to unexpected traffic events as it eliminates the influence of events by averaging historical observations. ARIMA, due to its delayed reaction, is not an ideal method to use in the case of events which cause sudden changes in the time series data”.

A. *Event Impact Area Model.* Impact post-mile is the distance between the location of an event and its last influenced sensor on the opposite direction of vehicle flow, as shown in Figure 5.

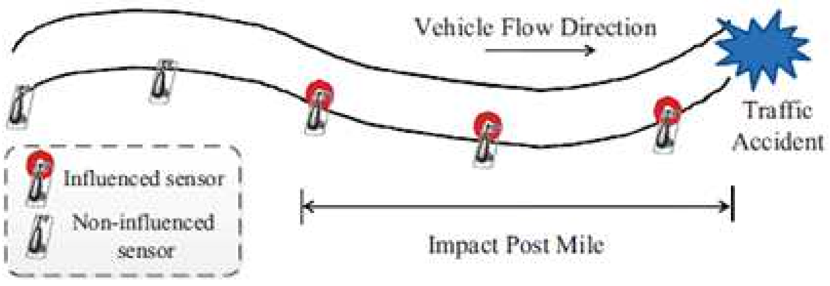


Figure 5.- Definition of event impact post-mile

the following event attributes are the most relevant:*{Start time*, *Location*,*Direction*, *Type*, *Affected Lanes}*.

B. Event Impact Prediction.

For sensor *i*, its influence speed decrease Δv_i for event *e* is defined as the average speed changes for all events that share the same correlated attributes (i.e., *Starttime*, *Location*, *Direction*, *Type* and *Affected Lanes*) with *e*, and affected sensor *i* in the past.

For sensor *i*, its *influenced time shift* (Δt_i) for event *e* is defined as the distance between the sensor *I* and event *e* divided by the average traffic speed between them, which can be represented as follows:

$\Delta t_i(e) = dist(i, e)/avg(\{v_j\})$ where $p(i) \leq p(j) \leq p(e)$

where *p(i)* refers to the post-mile of sensor *i*. The set of$\{v_j\}$ refers to all the speed readings presented at the sensors located between sensor *i* and event *e*.

1) When an event *e* occurs at time *t*, all the relevant event features(i.e., *{Start-time*,*Location*, *Direction*, *Type*, *Affected Lanes}*) are incorporated in the EIA model to determine the impact post-mile of *e*.

2) Using the impact post-mile and the location of *e*, the set of all influenced sensors are identified as set $\{s_i\}$.

3) For each sensor s_i, during $[t+\Delta t_i(e), t+\Delta t_i(e)+h]$, the predicted value is calculated as $(v_i(t) - \Delta v_i)$, where h is the prediction horizon.

4) After time $t+\Delta t_i\ (e)+h$, ARIMA is used to predict the rest until the event e is cleared.

Performance evaluation.

Baseline Approaches:

ARIMA: "*We implement ARIMA starting with stationary verification*", followed by the iterations of 1 to 10 for Auto Regressive model and 1 to 10 for Moving Average model to reach the best combination under "*Bayesian information criteria*". We use the trained model for one-step (h = 1) forecasting. When h > 1 (i.e., long-term forecasting), we iterate the prediction procedure for h times by using predicted value as previously observed value.

ES: We implement Exponential Smoothing (ES) method as a special case of ARIMA model, with the order auto-regressive model set to zero, and the order moving average model set to 2.

NNet: We implement Neural Network (NNet) model as multilayer perceptron (MLP). The architecture of MLP is as follows: 10 neurons in the input layer, single hidden layer with 4 neurons and h output neuron, where h refers to the prediction horizon. For example, in onestep forecasting, there is 1 output neuron. The input neurons Include $\{v(k), k = t - 9, ..., t\}$, while the output neuron is $\{v(t+1)...v(t+h)\}$, where t represents the current time. Tangent sigmoid function and linear transfer function are used for activation function in the hidden layer and output layer, respectively. This model is trained using back-propagation algorithm over the training dataset.

Fitness Measurements: We use mean absolute percent error (MAPE) and root mean square error (RMSE) to quantify the accuracy of traffic prediction.

$$\text{MAPE} = \left(\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \widehat{y}_i|}{y_i}\right) \times 100$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \widehat{y}_i)^2}$$

where y_i and $\widehat{y_i}$ represent actual and predicted traffic speed respectively, and *n* represents the number of predictions.

Predictions without event information.

Short term prediction.

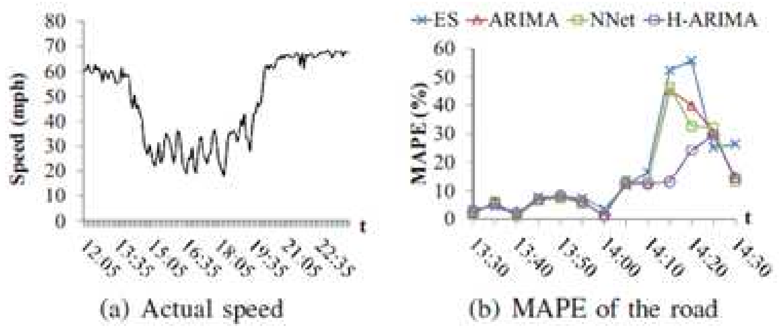


Figure 6.- Case study on I-5S. segment from Downtown.

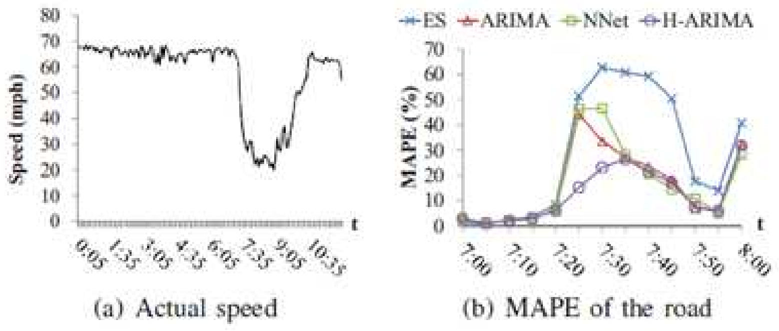


Figure 7.- Case study on I-10W. segment to West-LA.

We can observe in figure 6b, 7b, that Hybrid ARIMA (H-ARIMA) outperforms the other contenders significantly. However we must notice that at peak 7.25 and 7.30 the neural network recovers its performance, this means learns, to predict the traffic flow very fast in 10 minutes and performs very close to H-ARIMA. IN both figure it is reflected the learning effect that a neural network has. Long term prediction:

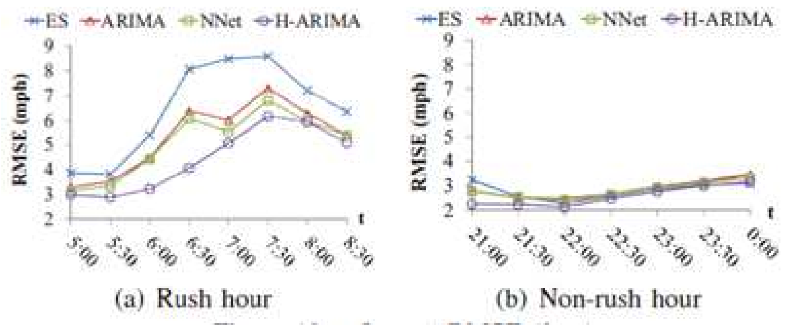


Figure 8.- Overall RMSE (h=6).

When there are sudden changes in traffic flow at rush hours boundaries H-ARIMA outperforms its contenders. However Neural Networks follows very closely, 2nd best, the H-ARIMA algorithm.

As shown in Figure 11(a), around 7:00AM, the speed decreases from 65 mph to 5 mph within a very short time. The baseline approaches can only sense this change with 30 minutes delay, and hence their MAPE is considerably high (see Figure 9(b)). On the other hand, H-ARIMA utilizes the historical congestion information to predict the traffic and hence its MAPE is fairly low as compared to baseline approaches. This delay is also due to the time needed for the NNetwork to learn, However if we are able to introduce historical congestion information into the neural network this could open an opportunity for neural networks. Something like having two neural networks one for prediction at rush hours and the other for prediction at non-rush hours.

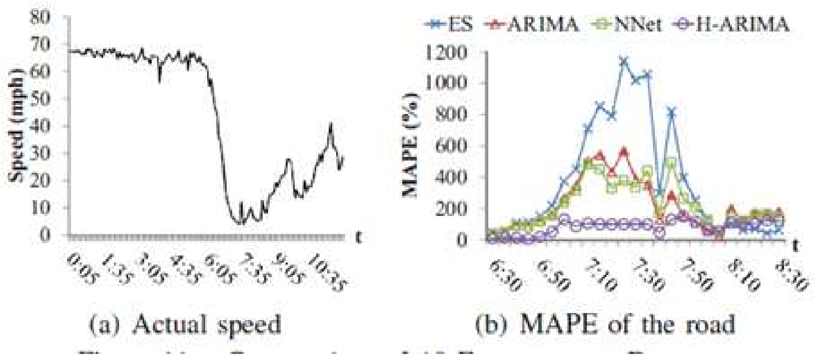


Figure 9.- Case study on I-10 E. segment to Downtown.

Predictions with event information.

In figure 10, we can observe the predictions of NNetworks, H-ARIMA, and H-ARIMA+ (the last one depicted in EVENT IMPACT ANALYSIS) for three traffic accidents. H-ARIMA+ outperforms its contenders. Neural networks take around two hours to be competitive with respect to H-ARIMA+ predictions.

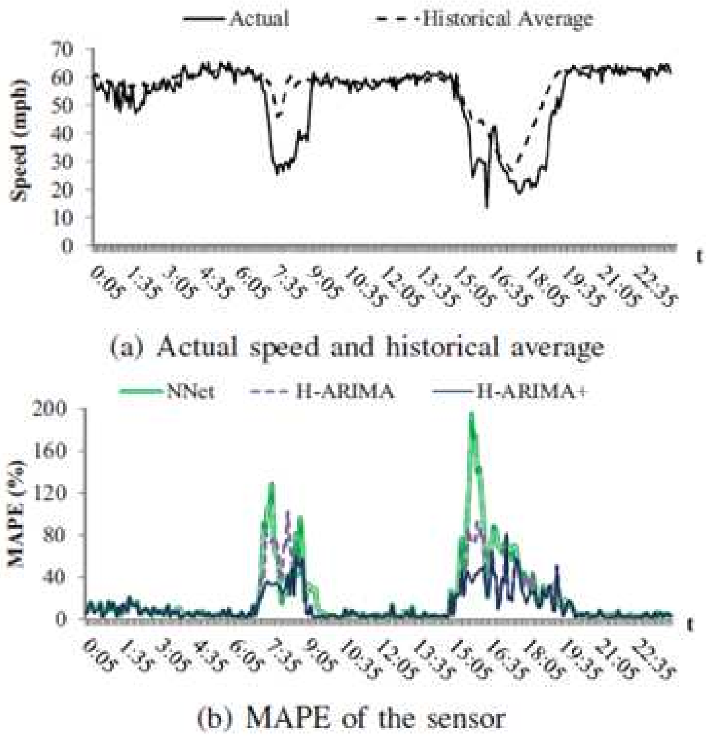


Figure 10.- Case study on traffic collision events.

In figure 11, a road construction event happens and the prediction speeds are plotted. Again H-ARIMA+ beats its contenders and neural networks take about one hour to learn to predict traffic, or only does it well because average speed return near to historical data (training data).

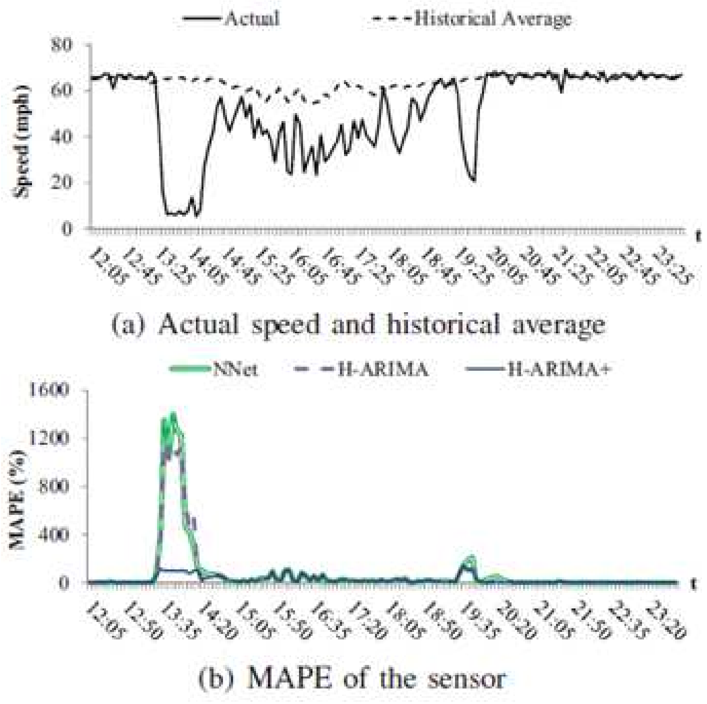


Figure 11.- Case study on road construction event.

Areas of opportunity:

- temporal patterns during rush hours, or
- spatial impacts of accidents
- Feature selection for correlated events during traffic flows.
- Training and learning model for neural network
- Introduce historical information about congestion
- Use two neural networks for traffic prediction.
- A neural network that takes acceleration instead of speed as medium to predict speed. And take samples more often i.e. every minute instead of every 5 minutes.
- Taking second derivative to predict speed. A mean to sensor the sense of direction change to predict traffic speed.

2.4. Traffic flow data mining based on cloud computing.

From [9], we can get the following information and questions of relevance:

Traffic data mining.

- Cloud computing can help to handle the large amount of storage resources and mass traffic flow data effectively and efficiently how?
- New architecture
- Data mining is to extract patterns from large data sets by combining methods from statistics and artificial intelligence with database management
- the traffic data includes the traffic management control data, Road environmental data, and road traffic data. common data mining algorithms cannot meet the specific application requirements of ITS for traffic data mining analysis.
- Traffic data is of time characteristics and strong spatial correlation.

Traffic flow forecasting.

The traffic flow forecasting methods can be roughly divided into four categories: the methods based on traditional statistical theory, the methods based on nonlinear theory, the methods based on knowledge discovery and the methods based on emerging technologies such as data fusion.

These methods have advantages and disadvantages and conditions. The algorithms based on the traditional methods of statistical theory are relatively simple. However, when the prediction interval is less than 5 minutes, the randomness and non-linear of the traffic flow changes will become strong, so the precision of the model will become poor. The algorithms based on nonlinear theory are well suited to the uncertainty of the changes in the traffic flow, but the calculation is more complicated, and the theory is not yet mature. Knowledge discovery-based approaches are good at identifying the characteristics of complex nonlinear systems, but they require a lot of training programs in the learning phase.

In summary, the complex features and limitations of the prediction model of the traffic flow under different conditions are difficult to obtain an accurate prediction results by the use of a single model and method.

The artificial neural network (ANN) when more training samples, the training time is too long and easy to fall into local minima.

The rough set (RS) theory is a new knowledge acquisition method. It is capable of expressing and processing incomplete uncertain traffic information. By the indiscernible relation reduction, RS can extract information between potential rules, and obtain the minimum expression of knowledge. At the same time, the algorithm is simple and easy to operate. However, in practical applications, RS theory is noise sensitive and poor at anti-interference ability. It can be found through the analysis that ANN and RS theory as two methods of data processing, there are many complementarities. In our research, RS and ANN are fully integrated with the advantages of both to build traffic flow forecasting model. In our model, we use RS theory to pre-process the traffic data, eliminate the redundant data and attributes and extract the rules. Then, we use the rules to determine the number of hidden layers, hidden layer nodes and connection weights of the artificial neural network to complete the network structures. Finally, the data after reduction will be input to the artificial neural network to predict traffic flow. The model has a good transparency, and is easy to explain, and is of good generalization ability and anti-interference ability. In our approach, the estimation results of the traffic flow are show as Fig. 12. As the figure indicates, the combination of ANN and RS has got more accurate estimation. The architecture of the cloud is divided into: user interaction interface, system resource management module with a services catalogue, and resource provisioning module. Cloud computing system can be divided into two sections: the front end and the back end. Cloud architecture has five layers. The results shows that the cloud computing can greatly enhance the efficiency of the data mining:

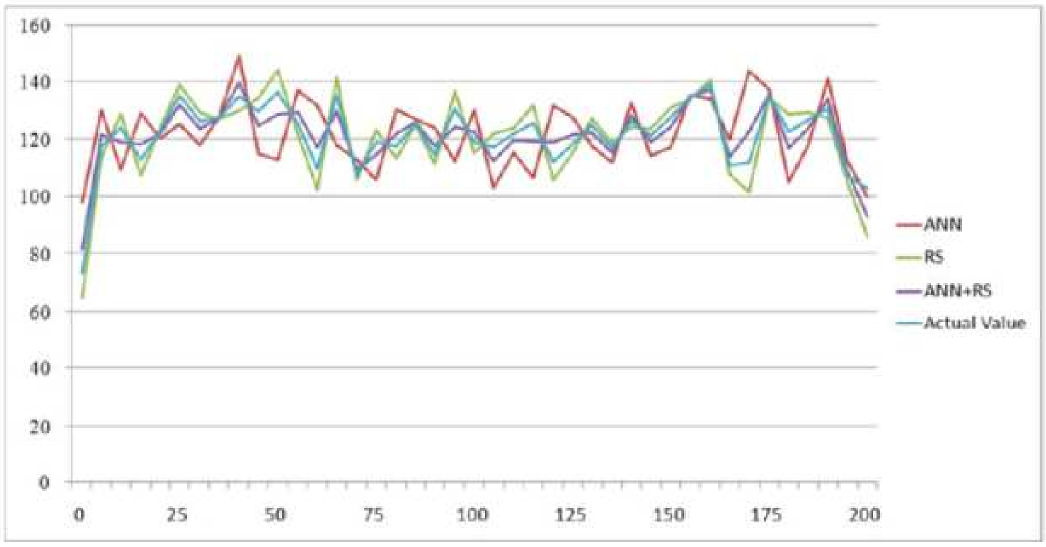


Figure 12.- The estimation results and the actual values of the traffic flow.

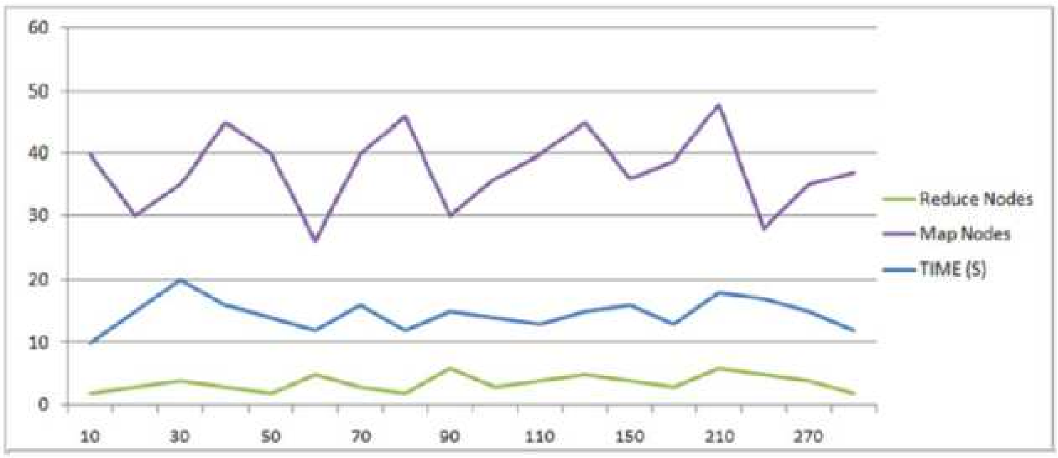


Figure 13.- Results of different Reduce Nodes and Map nodes.

Areas of opportunity:

- Does not show error measurements, so we have no idea how close to real data are the predicted values.
- Rough theory is something that is not mentioned by several article writers.

2.5. Real time traffic forecast.

To articulate a PBS (Pattern Based Strategy) practice, several knowledge disciplines need to be considered, such as predictive analytics, data mining, complex event processing, business intelligence, big data, decision support, social network analysis, etc.

Participants and partners

Several partners have participated in the Real Time Traffic Forecast (RTTF) PoC:

- Atos Scientific Community (PBS track) to define the principles of Pattern-Based Strategy to be applied in the PoC (Proof-of-Concept);
- Atos C-LAB as project leader and responsible of the system and solution design, solution development, integration and implementation;
- Siemens Corporate Technology providing a solution kernel with a forecasting model, developed using their widely proved Neuron Network-based system SENECA;

- VMZ Berlin (Traffic Management Central for the city of Berlin) as users of the final system. They provided historical data from their city sensor infrastructure and supported the project with their profound domain knowledge.

Data acquisition and processing.

Initial data collection. Acquired data from loop detectors and infrared sensors every 60 seconds during summer months.

Data cleansing. This is to avoid unwanted information resulting from missing or corrupted data due to ie malfunctioning sensors. This was done by filtering and interpolating values when necessary.

Data usage. The data was divided in two samples:

- Modelling sample: 60 days for pattern forecasting.
- Control sample: 30 days of data to simulate real time traffic and to test system prediction.

Pattern identification and modeling.

Domain expert support: Traffic status determination.

Two variables were taken into account speed of cars and traffic flow rate which determines the number of vehicles passing a traffic sensor per hour. Other parameters were road capacity, and speed limits, etc. The model created was validated by a domain expert. Using a look-up table the traffic model provides the traffic status which is divided into four categories: free flow, slow moving, congestion and undefined.

Data scientist: Artificial neural network (ANN).

Data feed.

We have 100 sensors, with a prediction horizon of 4 hours, with an update period of 5 minutes. There is two sampling vectors, one for speed and one for flow rate, containing the value of four hour data stream with sampling rate of 1 minute this leads to an entry vector of 240 entries for each sensor.

Data output: Two output vector containing the forecast for the next 4 hours, for speed and flow rate.

Training and results: The ANN was trained with back propagation method. The predictions shown in the following figure are in blue while the real time sensors data is in green. In order to quantify the forecast quality it was averaged the deviation of 100 sensors over a 30 day time period revealing 93% precision of traffic speed and 86% for flow rate. Keeping in mind that a 60 day period is quite short and no additional information eg social events, road closures, etc has been fed into the ANN the results are promising.

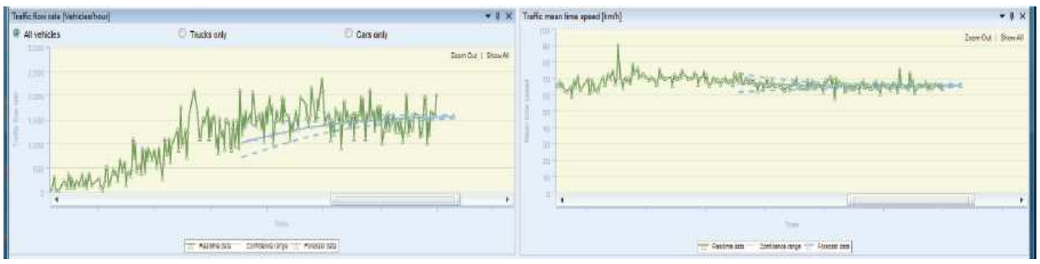


Figure 14.- Neural Network Estimation.

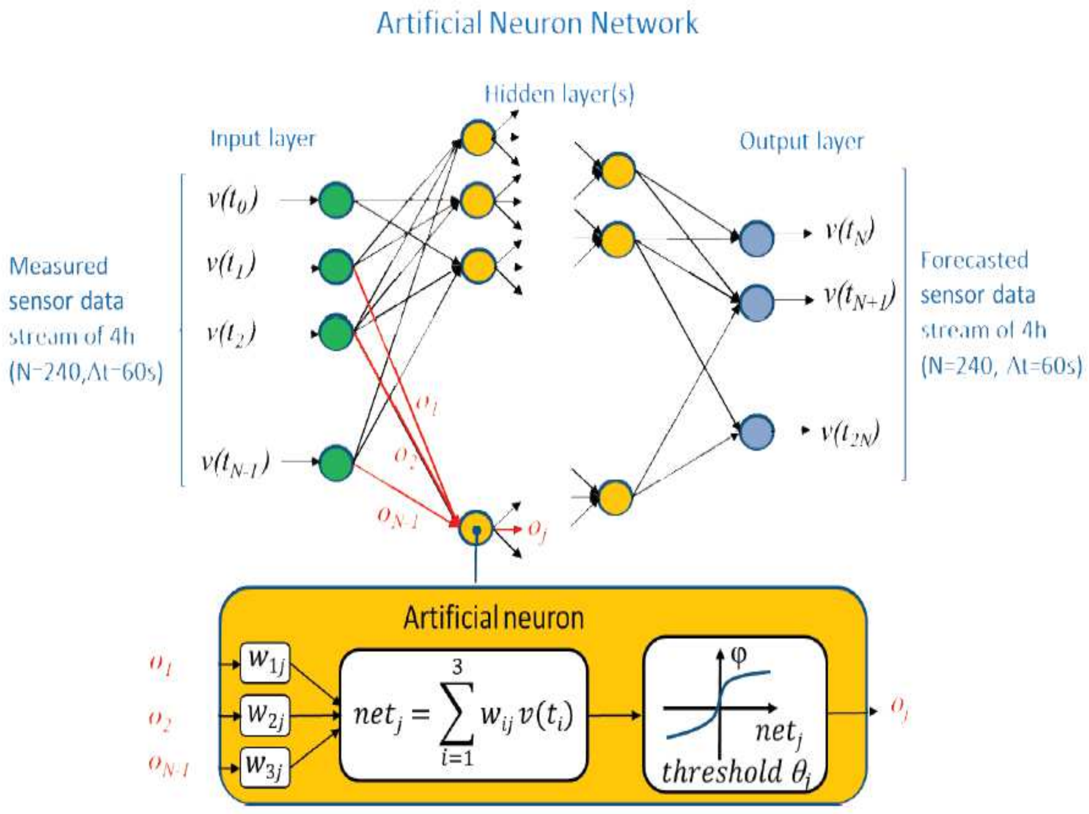


Figure 15.- Structure of the neural network.

In the following figure it is depicted how the distributed system was deployed in order to put under test its capabilities. It consists of a Traffic Management Server (TMS), Traffic Forecast Server (TFS), and a traffic monitor client (TMC).

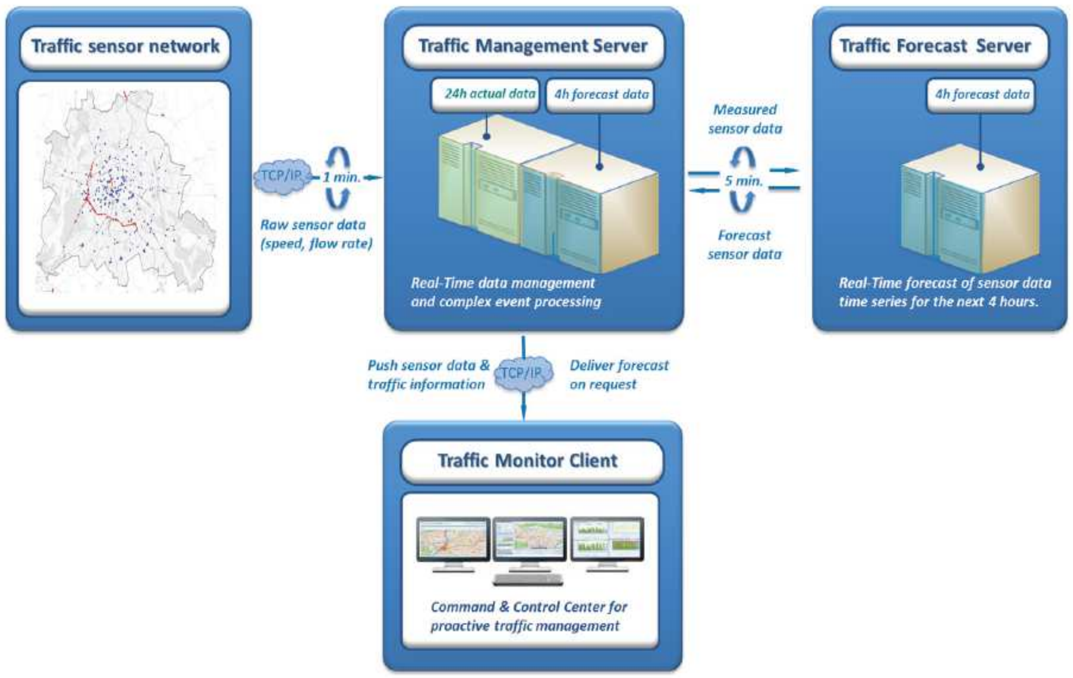

Traffic sensor network
TCP/IP
1 min.
Raw sensor data (speed, flow rate)
Traffic Management Server
24h actual data
4h forecast data
Real-Time data management and complex event processing
Measured sensor data
5 min.
Forecast sensor data
Traffic Forecast Server
4h forecast data
Real-Time forecast of sensor data time series for the next 4 hours.
Push sensor data & traffic information
TCP/IP
Deliver forecast on request
Traffic Monitor Client
Command & Control Center for proactive traffic management

Figure 16.- Structure of the system for traffic prediction.

For the PoC it was created a simulator of real time streaming data every minute. In the TMS it is stored the simulated real time data for the last 24 hours. Every 5 minutes the TMS pushes a real time traffic data update into the TFS which contains the ANN. The speed and flow rate are predicted for the 100 sensors at the next 4 hours with a sampling rate of 1 minute. The forecasted data is stored back to the TMS. After this the TMC is updated with the new traffic forecast.

Areas of opportunity:

- How did they filter and interpolate the data?
- Scheme definition of the neural network.
- Which was the total set of parameters used for constructing the traffic model.
- Add social events, road closures, to ANN training.
- Computation done by ANN must be done in less that one minute, the sampling rate.

2.6. LA department of Transportation(LADOT).

For the purpose of this thesis and the purpose of comparing our results with previous published results, we are going to use the traffic database from the department of transportation of california at Los Angeles. The system is called the California Performance Measurment System (PeMS).

For general knowledge We have to mention that acquiring a permission to access the database called PEMS it took us about 5 months due to a very slow process. Once we obtained the permission the next step was to test the connection because at ITESO ftp connections are prohibited by default. The first trial was unsucessful and we have to register for a new IP LADOT, this took us one more month. Once we got the new IP registered we have to test again. This time it worked. And now we'll have to understand how the PEMS database work. We found out that the ftp server to which we have to connect and download the information only stores the last 35 minutes of traffic information in text files created every 5 minutes.

We created an ftp client that connects to the PEMS database and is in charge of retrieving the last 30 minutes of information available at the PEMS ftp server. The connection is depicted in the figure 17.

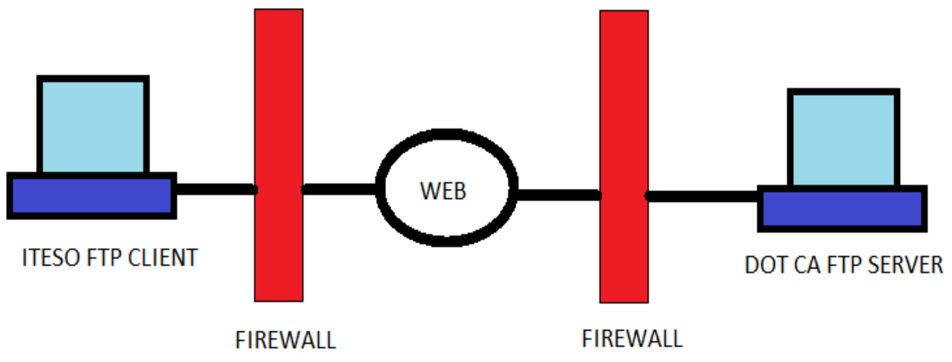


Figure 17.- Internet connection between ITESO and CALTRANS.

The FTP client.

The ftp client is mounted on a Linux machine with ubuntu 14.04. It runs in a periodical way thanks to a CRON job. The CRON job is configured to run every 30 minutes, every day of the week, and month, at all hours. If something goes wrong it sends an email to a local inbox that can be consulted by running *mutt* the mail client.

The architecture of the FTPClient is depicted in the next figure:

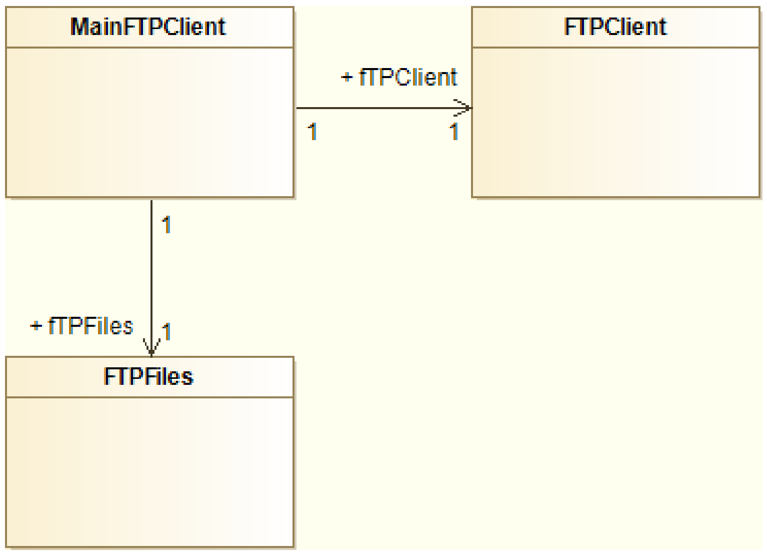


Figure 18.- Class diagram for the ftp client.

The FTP client algorithm is as follows:

1. Connect to FTPServer at LADOT.
2. Loggin into FTP server.
3. Change transfer to binary mode.
4. Enter passive mode.
5. Change to remote working directory.
6. List the remote files (FTPFile).
7. Download every remote file to a local directory.
8. Logout, disconnect, and close connection to FTP Server.

The text files with traffic information data.

The text files are called as 5minagg_20150421150500.txt.gz, so the first step is to unzip them and then analyse its contents. In the LADOT web page information (a private page that you have access only after having registered an IP). The format for this naming convention is: 5minagg_YearMonthDayHourMinuteSecond.txt.gz

The format of a txt file containing traffic information is as follows:

Table 1.- Information obtained from the CALTRANS metrics file.

Column	Description
VDS_ID	Unique station identifier
FLOW	Flow (vehicles/5-minutes)
OCCUPANCY	Average occupancy as a percentage (0 - 1)
SPEED	Flow-weighted average of lane speeds
VMT	Total vehicle miles traveled over this section of freeway
VHT	Total vehicle hours traveled over this section of freeway
Q	Measure of freeway quality (VMT/VHT)
TRAVEL_TIME	Not in use
DELAY	Vehicle hours of delay
NUM_SAMPLES	# of samples received in the 5-minute period
PCT_OBSERVED	Percentage of individual lane points from working detectors that were rolled into the station\\'s 5-minute values.

The XmlMetrics2DB.

The XmlMetrics2DB is a file that takes as input a vds station config element, and a text file with metrics information about traffic and sends this information to the MySQL database. See the following picture for a pictorial description:

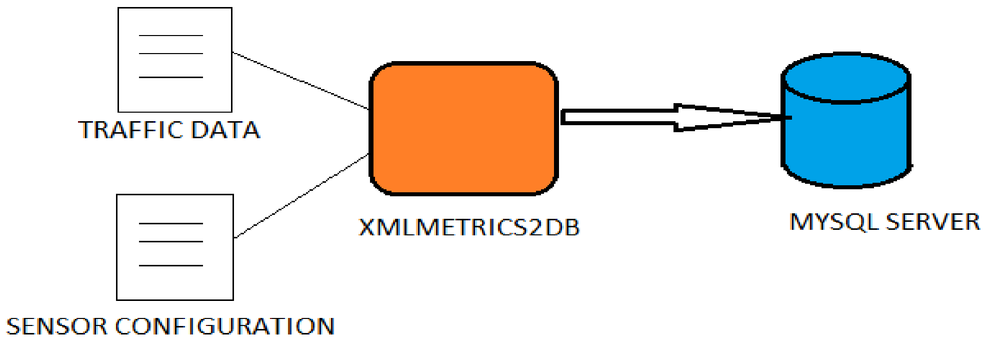


Figure 19.- Flow of information from unstructured data to MySQL database

The architecture of the programs is as follows:

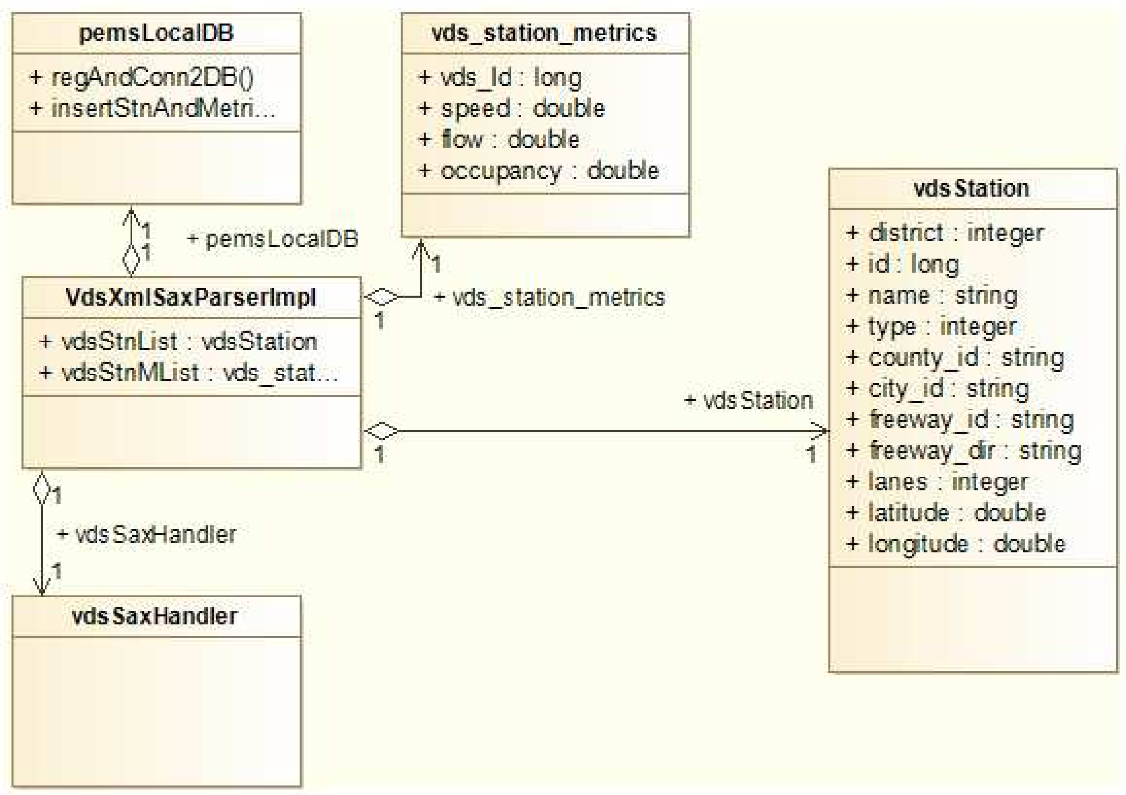


Figure 20.- Class diagram of the XML2DBMetrics program.

The algorithm is as follows:

1. First the vds_config.xml file is parsed and store in vdsStnList. This XML file contains the configuration of every station. The vdsStnList is sorted.
2. Registry and connect to the database.
3. Parse a single metrics file and for every station available in vdsStnList insert a row in the DB.

2.7. The neural network test.

Eventhough we still don't have the full set of data available for the neural network prediction. We use a subsample of it in order to see a preliminar behavior of the neural network.

First we consider the openloop training of the neural network. We use a neural network of one output and 10 hidden neurons, with 20 inputs for x(t), and 20 inputs for y(t), with the Levenberg-Marquardt training algorithm. Check the following figure for a visual depict of the ANN:

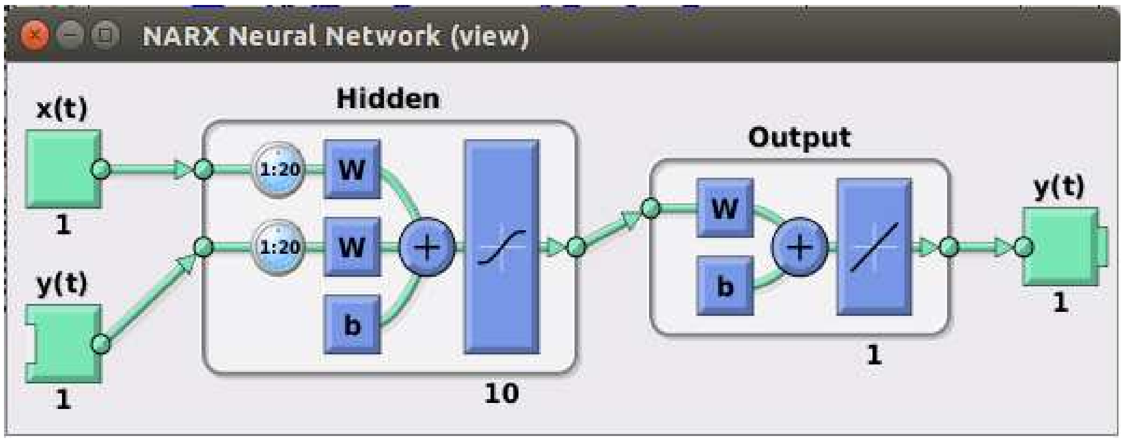


Figure 21.- NARX Neural Network.

The training response is the following:

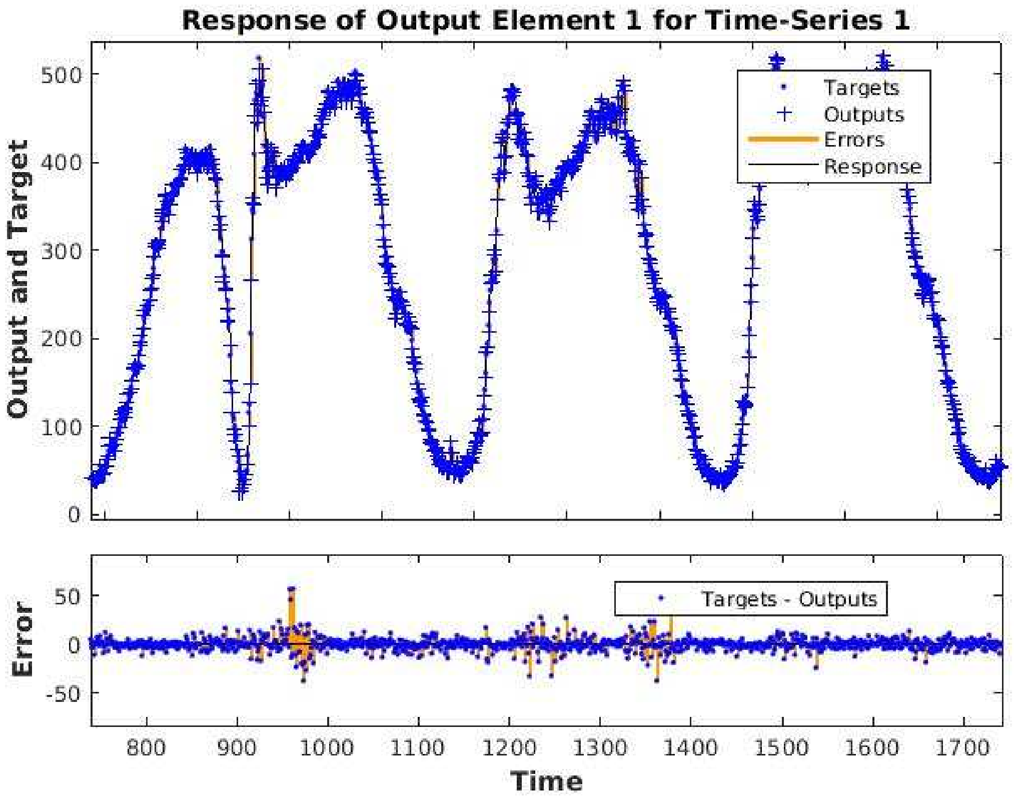


Figure 22.- Response of open loop ANN shown in fig 21.

In this figure we can observe the response of the training data as input vs training data as ouput. We can observe in this figure that the outputs (response of the ANN to new inputs) from the ANN are very close to the targets (values known from real data) of the ANN, thats why the errors are very close to zero around 5 vehicles per every 5 minutes of data flow. There are small peaks with value about 50, about 5 points, but the ANN recovers to minimal value of 5 very soon. This recoverage does not happen in previous results presented in this document, this recoverage last for 30 to 60 minutes, and the errors are about 120% to 200% vs our 1% obtained with Levenberg-Marquardt algorithm.

Now lets see a response from the ANN but with not known data by the ANN, we called this set the test data, this will be inputs/outputs not used for training the ANN. We can observe how the estimated outputs of the neural network are very close to the observed targets of real traffic data (targets). We can observe that the error has a value around zero and a few peaks, 5, about 20 to 50 vehicles per 5 minutes. Look at the following picture for details.

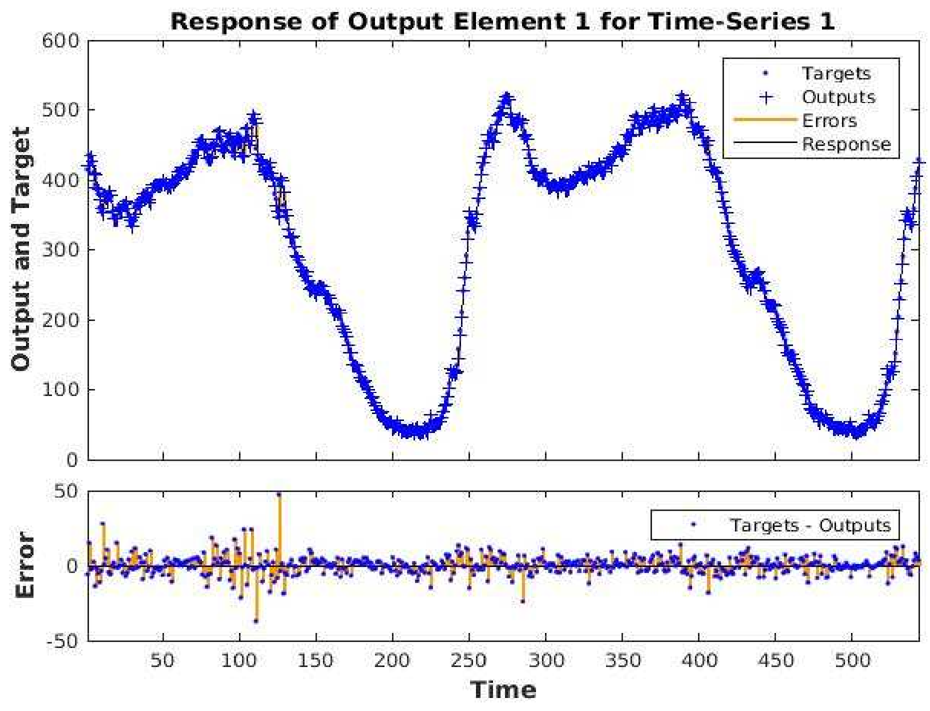


Figure 23.- Response of ANN shown in fig 21., but with not known data.

Now let's see the results obtained in close loop of the ANN. The following picture depicts the ANN (see figure 24).

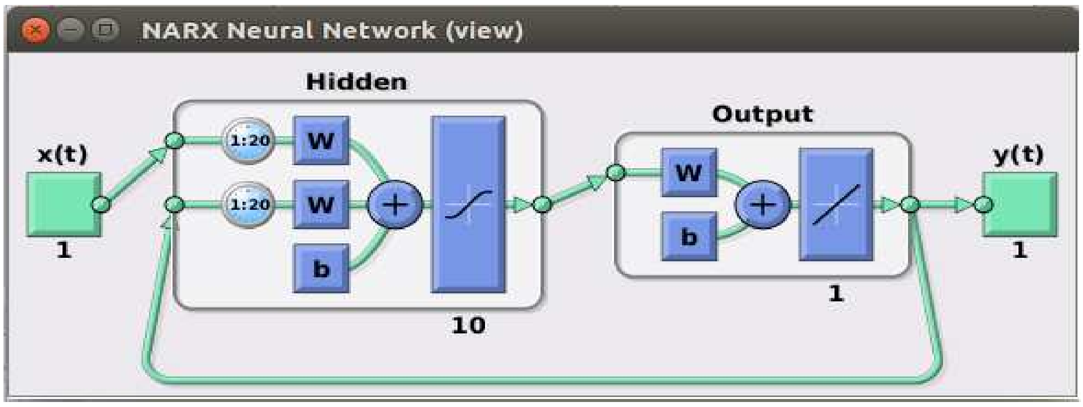


Figure 24.- Closed loop ANN.

Now let's analyze the response of the ANN in closedloop. We can see that the first 30 samples are very close to the real behavior however, since this is a closed loop aproximation the error at the output is feedbacked to the input of the ANN and this accumulates up to the point of having a great deviation from real data.

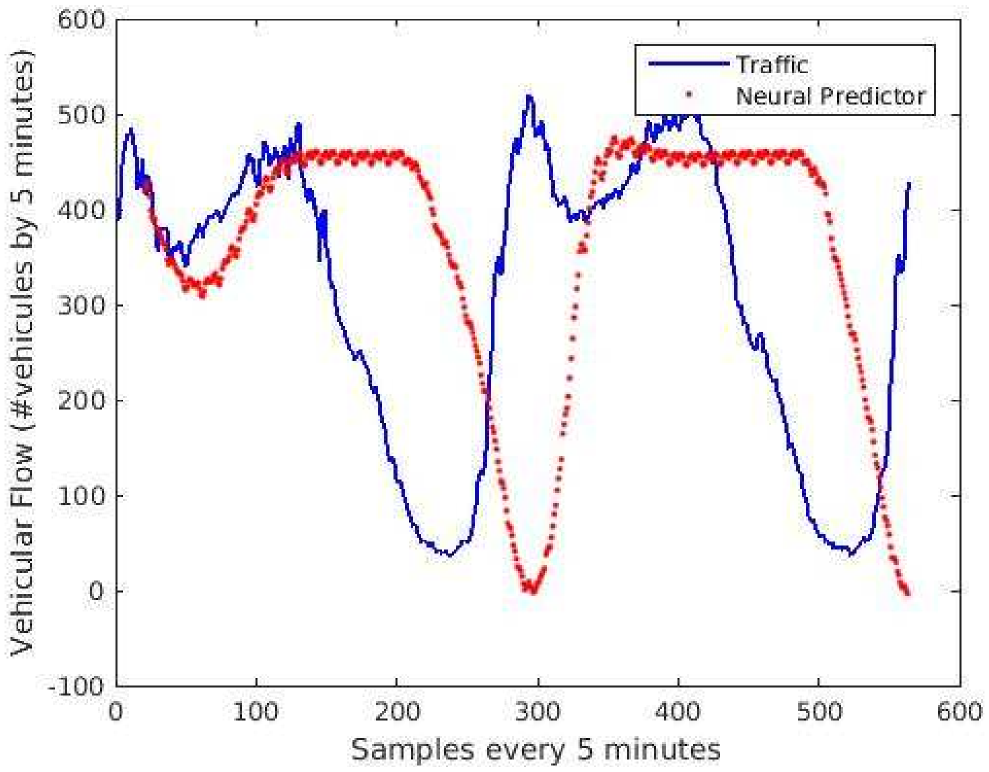


Figure 25.- Response of the closed loop ANN shown in fig. 24.

Now lets look at the ANN but with a delay removed in order to make it a predictor of one step ahead.

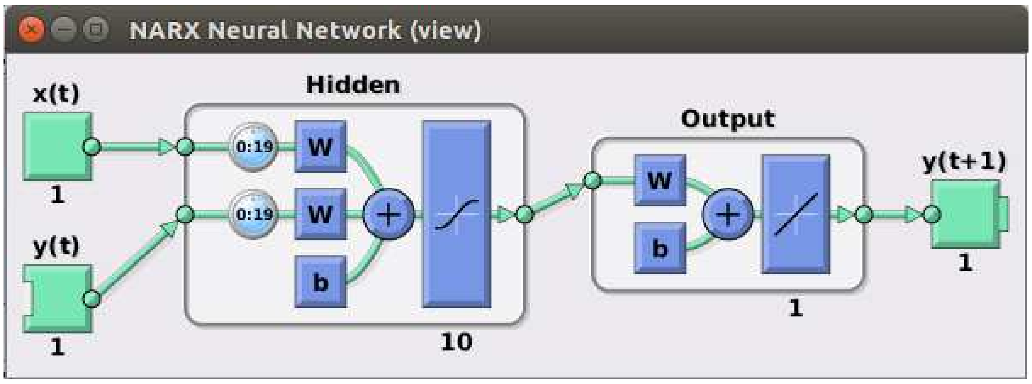


Figure 26.- One step predictor (predictor of 5 minutes ahead) ANN.

The response of this ANN is very close to the real data obtained for the traffic in LA, Notice this because the red dots that represent the estimated flow of vehicles, are following the blue line, the real traffic data, very closely.

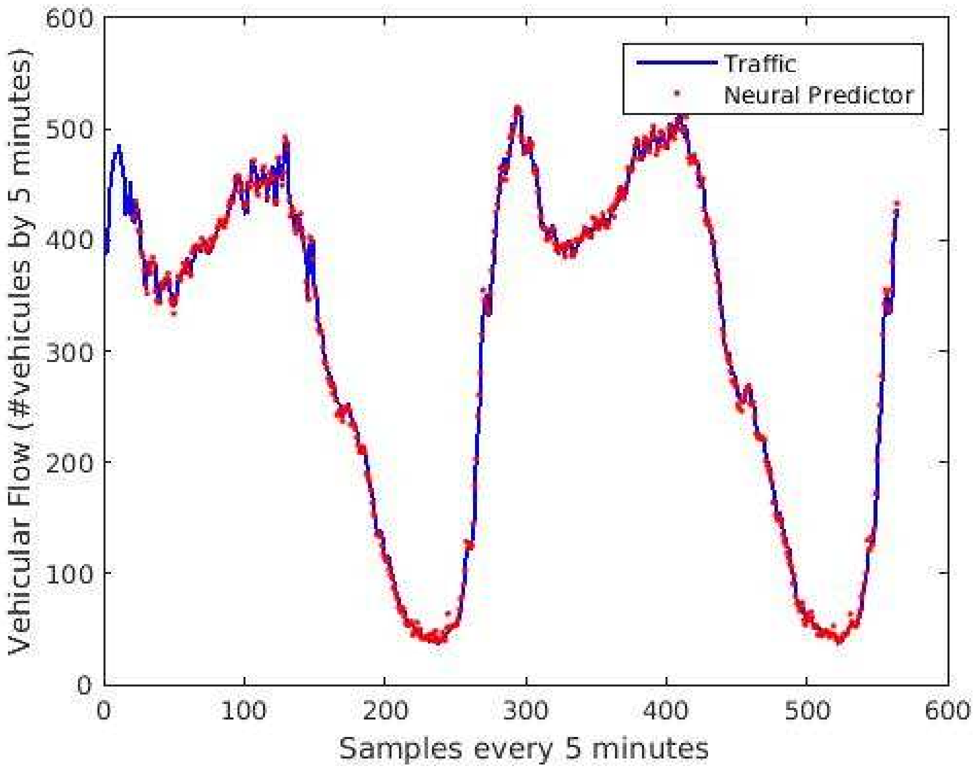


Figure 27.- Response of the one step predictor shown in fig 26.

2.8. *Prediction analysis with 3 months of data.*

In order to have a measurement that agrees with research literature we have to take 3 months of data from the freeways, in our case it is 95 days in order to be exact. Additionally, we cannot use anymore the previous NARX network tool from MATLAB because it cannot have multitple output, and it is of our interest to analyse a neural network with multiple ouputs that represent 30 minutes prediction of freeway flow in steps of 5 minutes; that's why we'll use a feedforward neural network tool from MATLAB to accomplish this objective.

2.8.1. *The feedforward neural net.*

We start our analysis of the feedforward neural network using as a basis the ANN shown in fig. 23, the NARX neural network. Notice that it has 20 delays for x(t) and another 20 delays for y(t). It also has a 10 neurons hidden layer, a nonlinear output of the hidden layer, and an output layer with 6 neurons representing everyone of them 5 minutes prediction starting from 5 minutes ahead up to 10,15,20,25, and 30 minutes prediction. The learning algorithm is Levenberg-Marquardt which is the default for the feedforward ANN. We chose this learning algorithm because it guarantees to reach a global optimum, in this case a global minimum error.

This details can easily be observed in figure 30.

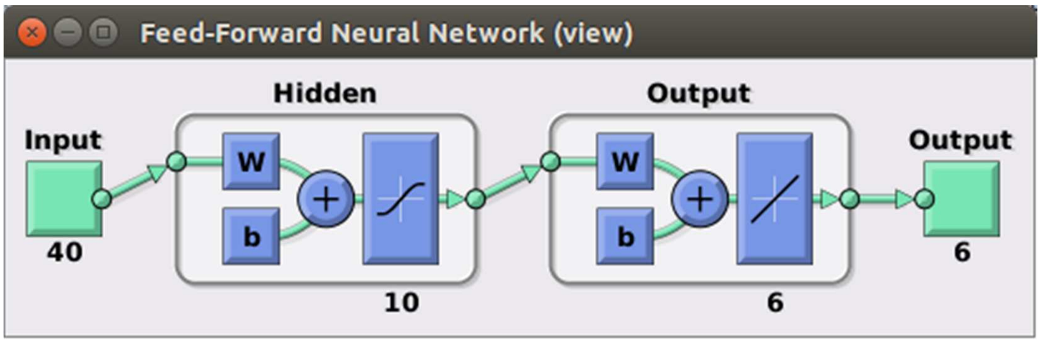


Figure 28.- Feedforward neural network equivalent to NARX ANN in fig. 21

2.8.2. *Response of the six outputs.*

In figure 31, we can observe the outputs of the feedforward neural network in red dots which are plotted against the real measurements of the freeways flow. The freeway flow (in vehicles per every 5 minutes) is plotted in y axis and sample number in x axis.

The outputs start from top to right and then bottom left to bottom right. This means that the 5 minutes ahead prediction is in top left corner, and 15 minutes ahead prediction is in top right corner, 20 minutes ahead prediction is in bottom left corner, and 30 minutes ahead prediction is bottom right corner.

Qualitatively speaking we can observe that the red dots, the output of the ANN, follow the blue line, the real measurements, very closely. This observation prevails for the six subfigures in figure 29.

Let's go deeper into a more quantitative measure to have an idea of the error of the feedforward ANN. In figure 30 the Root Mean Square Error (RMSE) goes from 8 to 17 vehicles per every five minutes. In figure 31 the Mean Absolute Percent Error goes from 2.5 to 5%, being this last one much better than the one observed in figure 16 of 86%, taking into account that these measures are for flow, speed will be analized further.

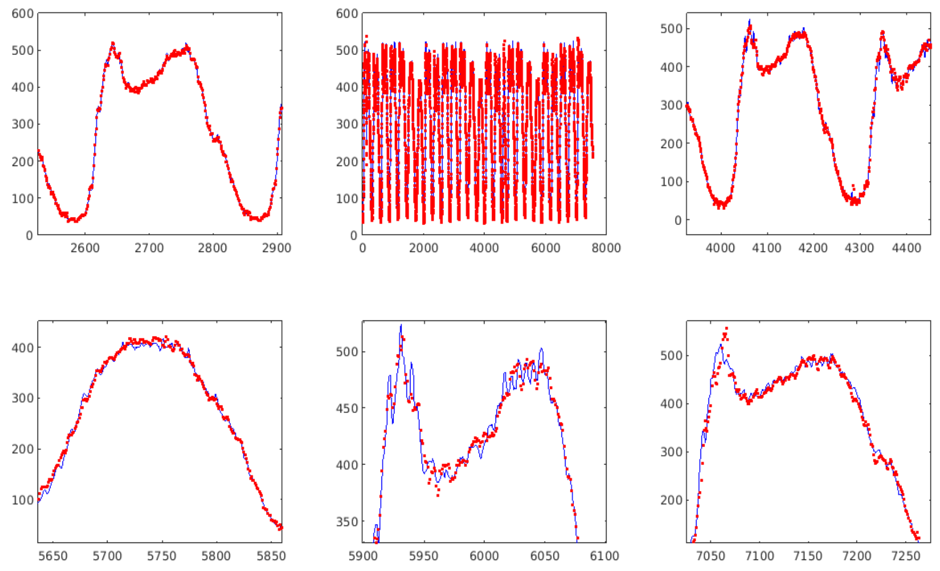


Figure 29.- Response of the six outputs (red dots) vs actual measurements (blue lines) for flow.

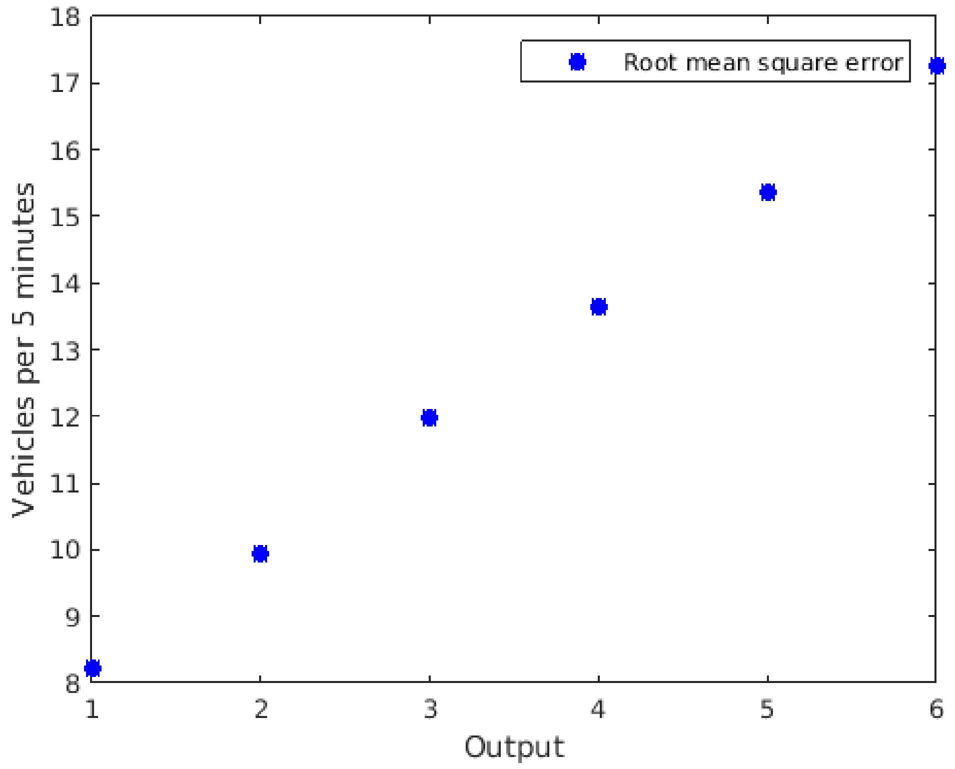


Figure 30.- Root Mean Square Error of every output for traffic flow.

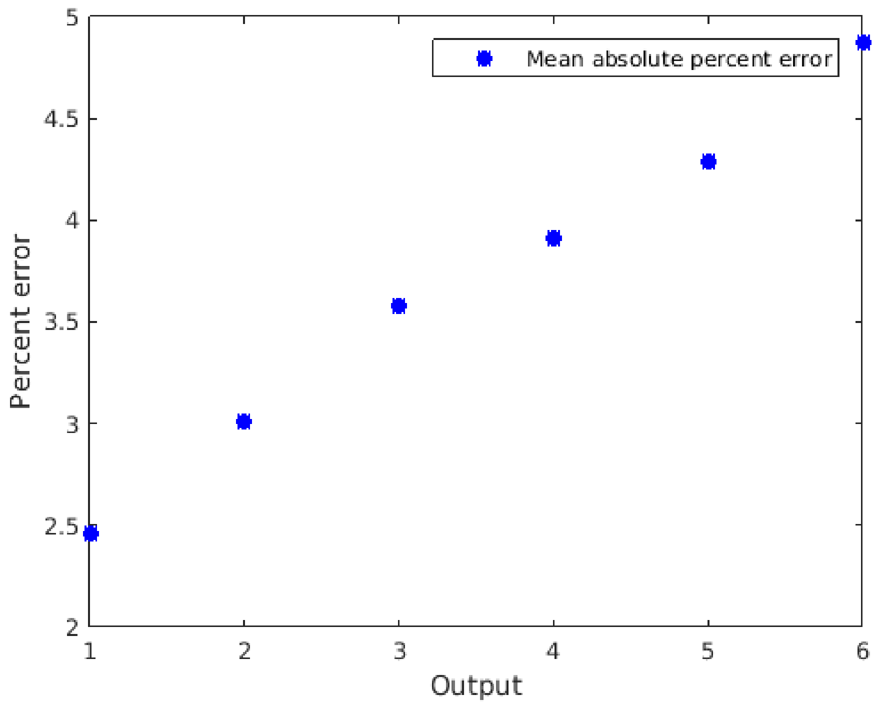


Figure 31.- Mean Absolute Percent Error for traffic flow.

2.8.3. *Optimizing the neural network for speed prediction.*

For the optimization process we calculate the Mean Square Error (mse) which is given by MATLAB after calling the function *perform* which calculates this index using the predicted output and the real observed output. We call it performance of the ANN and that's how it appears in the tables below.

In table 2, we can observe how the ANN was optimized to predict speed using 10 neurons in the hidden layer. We can observe how the minimun was obtained while the number of inputs was 18.

Table 2.- Traffic speed optimization using 10 neurons as the hidden layer.

Number of inputs	Number of delays in ANN	training_time seconds	performance	calc_time seconds
29	58	257.53	0.826	0.085779
26	52	311.98	0.8806	0.080749
23	46	185.46	0.9041	0.079706
20	40	180	0.863	0.13114
19	38	128.32	0.8105	0.077998
18	36	400	0.7876	0.143
17	34	138	0.8093	0.082463
16	32	313.99	0.9604	0.135426
14	28	261	1.1601	0.083092
11	22	101	0.8352	0.076305
10	20	103.295	0.8784	0.076864
9	18	55.68	0.8717	0.077719
8	16	32	0.9397	0.077948
5	10	119	0.935	0.099771
2	4	50	1.027	0.076067

In figure 34, we can observe a graphical depict of the of table 1, and how the minimum error, mse or mean square error, of the ANN was achieved at 18 inputs.

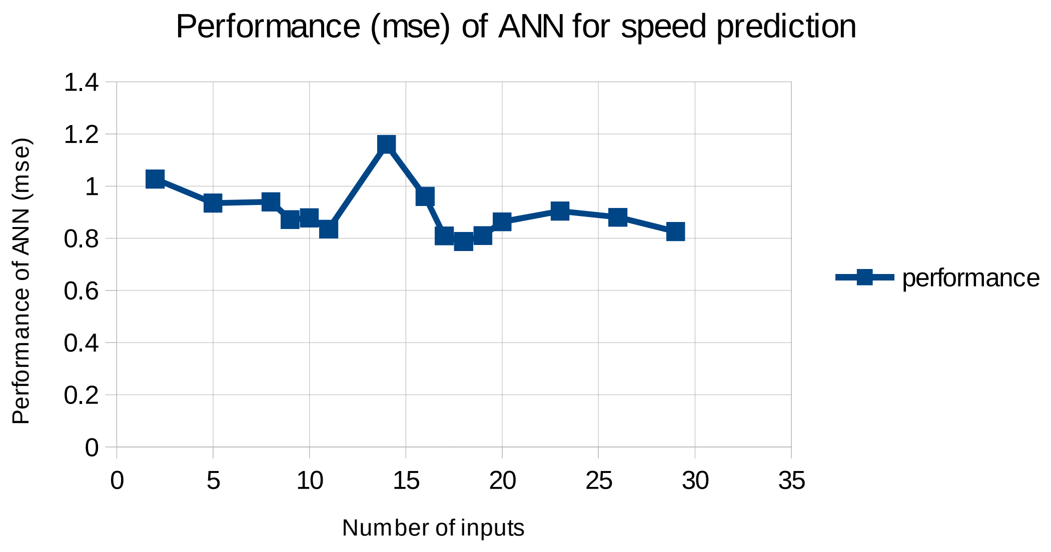


Figure 32.- Performance of ANN with 10 neurons in the hidden layer and predictions of speed.

In table 3 we can observe the optimization values for speed prediction in the ANN using 18 inputs, as a result from the table 1 and figure 32, but with different number of neurons in the hidden layer.

Table 3.- Traffic speed optimization using 18 inputs.

Number of neurons	training_time seconds	performance	calc_time seconds
20	354.182954	0.7789	0.173063
19	295.507273	0.7787	0.222018
18	411.110228	0.8343	0.175309
16	268.854945	0.9091	0.172757
14	90.643629	0.8466	0.173118
13	144	0.7546	0.185369
12	209.765779	0.7586	0.172804
10	400	0.7876	0.143
7	73.705001	1.0125	0.175178
4	101.023553	1.0843	0.172224

In figure 33, we can observe how the performance of the ANN reaches a minimum at 13 neurons in the hidden layer. From the previous two tables and figures we can observe that the speed prediction using an ANN reaches its minimum at the 18 number of inputs and 13 neurons in the hidden layer, this structure can be observed in figure 34. Notice that the input in the ANN is 36 due to the number of delays in the ANN being the double of the number of inputs.

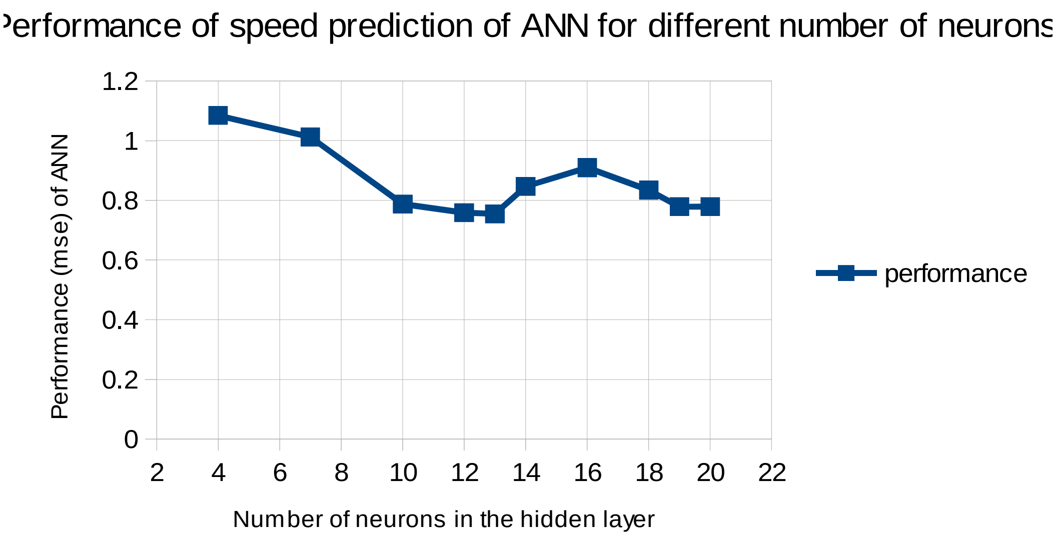


Figure 33.- Performance of ANN with 18 inputs and prediction of speed, with different number of neurons in the hidden layer.

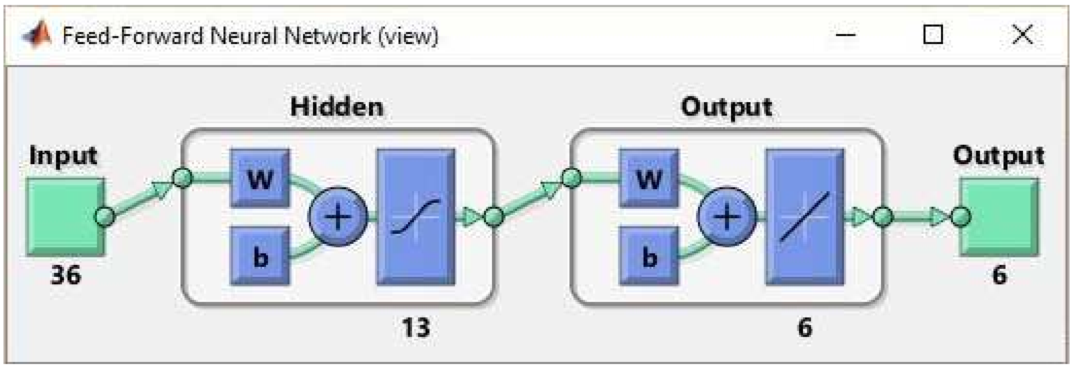


Figure 34.- ANN structure for optimal speed prediction.

In figure 35, we can observe the the Root Mean Square Error (RMSE) for the optimized ANN for speed prediction. It goes from 0.65 – 0.95 mph which compared to figure 8, whose values goes from 2 to 6 mph, it's a decrease that goes from 4 to 6 times.

In figure 36, we can observe the the Mean Absolute Percent Error (MAPE) for the optimized ANN for speed prediction. The MAPE goes from 0.57-0.95 % that compared with the values from figure 6b and 7b which are peaks that goes from 30 to 50% is a dramatical reduction in the error of 30 – 50 times less error.

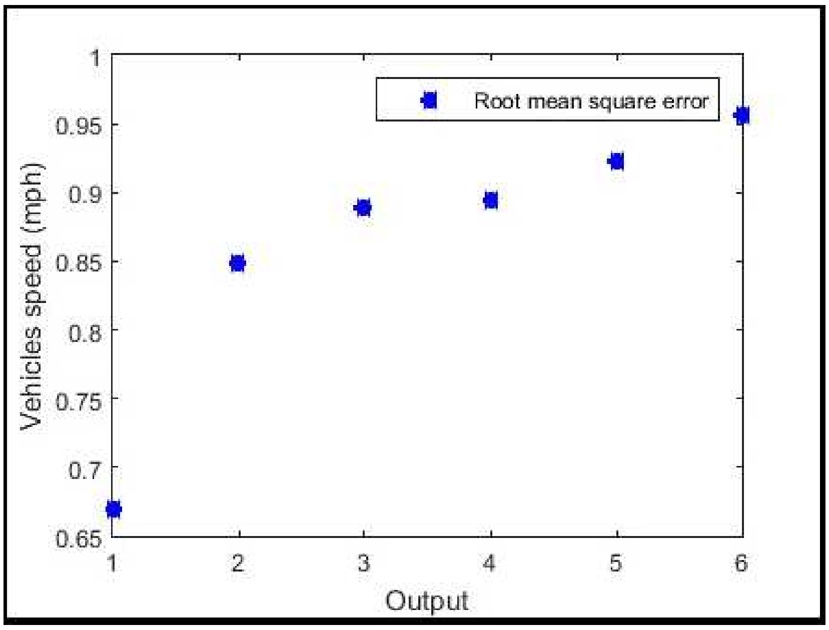


Figure 35.- Root Mean Square Error (RMSE) of every output of the optimized ANN for the speed prediction.

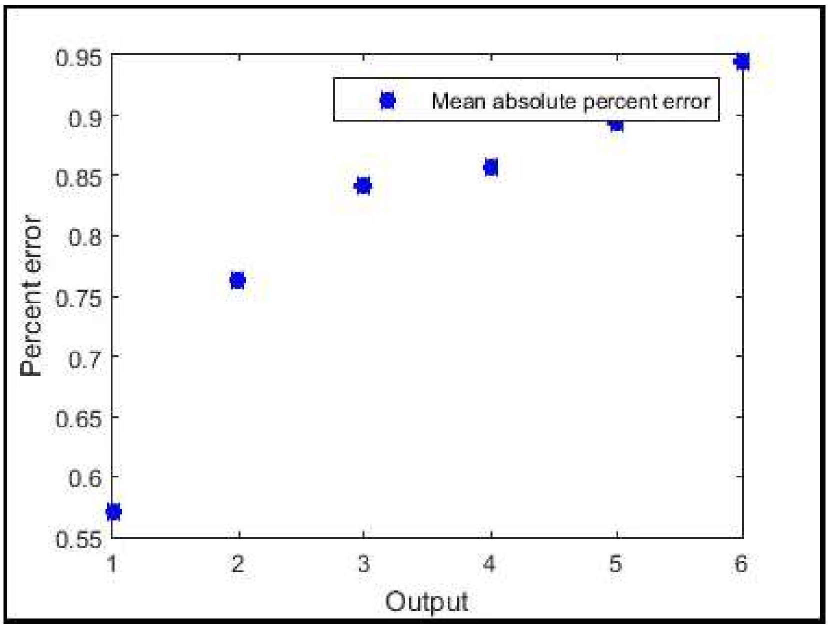


Figure 36.- Mean Absolute Percent Error (MAPE) of every output of the optimized ANN for speed prediction.

In figure 37, the outputs start from top to right and then bottom left to bottom right. This means that the 5 minutes ahead prediction is in top left corner, and 15 minutes ahead prediction is in top right corner, 20 minutes ahead prediction is in bottom left corner, and 30 minutes ahead prediction is bottom right corner. We can observe in this figure that the speed of vehicles has more peaks than the flow which makes it more difficult to predict, however, the results are much better than the observed in the literature.

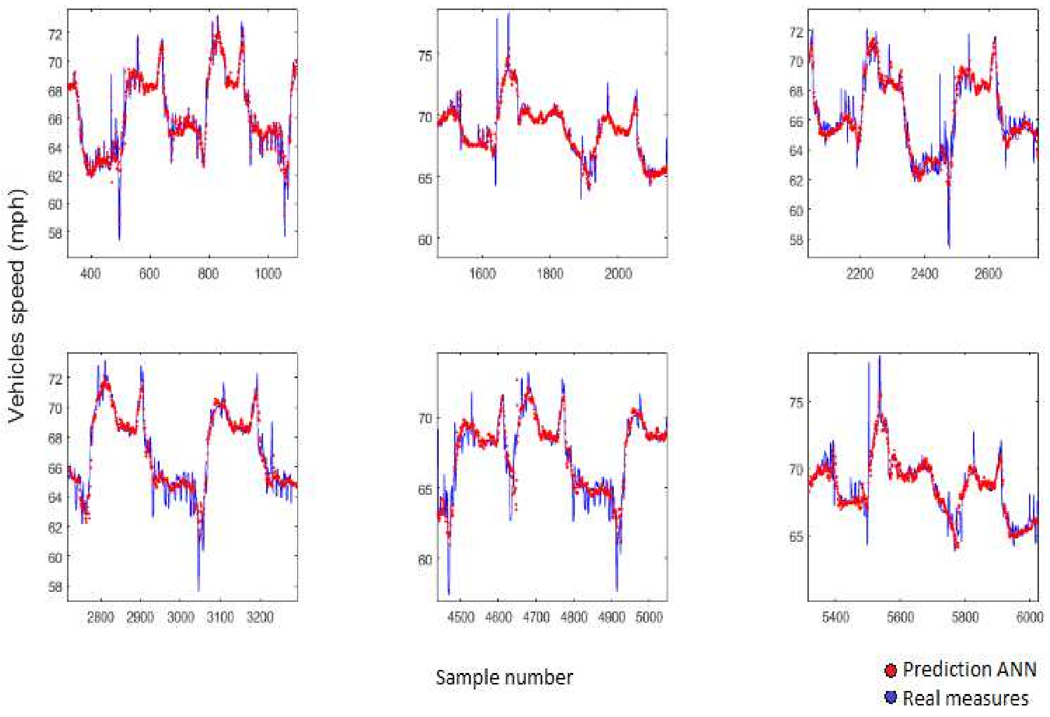


Figure 37.- Response for speed of the six outputs (30 minutes prediction)

2.8.4. Optimizing the neural network for flow prediction.

In table 4, we can observe the different set of values used to optimize the performance of the ANN for the flow of traffic using as initial value 10 neurons in the hidden layer. We can observe how in this table the performance for traffic flow prediction reaches a minimum at 14 inputs. In figure 38, we can observe this behavior pictorically.

Table 4.- Traffic flow optimization using 10 neurons as the hidden layer.

Number of inputs	Number of delays in ANN	training_time seconds	performance	calc_time seconds
26	52	210.419888	167.8752	0.114671
23	46	170.938954	144.1526	0.109144
20	40	98	155.2231	0.154616
17	34	120.37	177.3368	0.106406
15	30	177.753	142.1478	0.10732
14	28	88.98	139.0008	0.104202
13	26	79.45	154.4341	0.110264
11	22	74.2	167.4426	0.108857
8	16	96	246.9245	0.106848

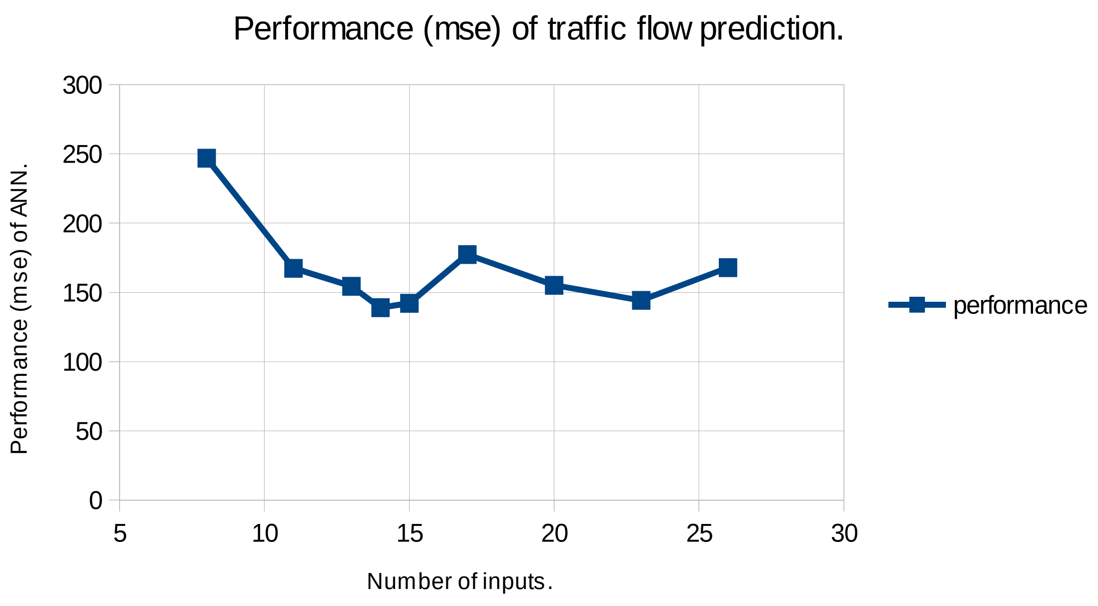


Figure 38.- Performance of ANN with 10 neurons in the hidden layer and prediction of traffi flow.

In table 5, using the results of table 3 and figure 38, we can observe the optimization process for flow prediction performance of the ANN using 14 inputs and varying the number of neurons in the hidden layer. We can observe in this table that the minimum performance is reached at the value of 16 neurons. We can observe this behavior pictorically in figure 41.

Table 5.- Optimization process for traffic flow prediction using 14 inputs.

Number of neurons	training_time seconds	performance	calc_time seconds
19	404.77784	197.1223	0.079653
17	532.111123	140.6576	0.078829
16	209.858048	105.347	0.081804
15	204.158633	107.4933	0.079259
13	180	164.8486	0.103343
10	88.98	139.0008	0.104202
7	68.613853	172.5722	0.077178
4	15.965465	210.7237	0.130687

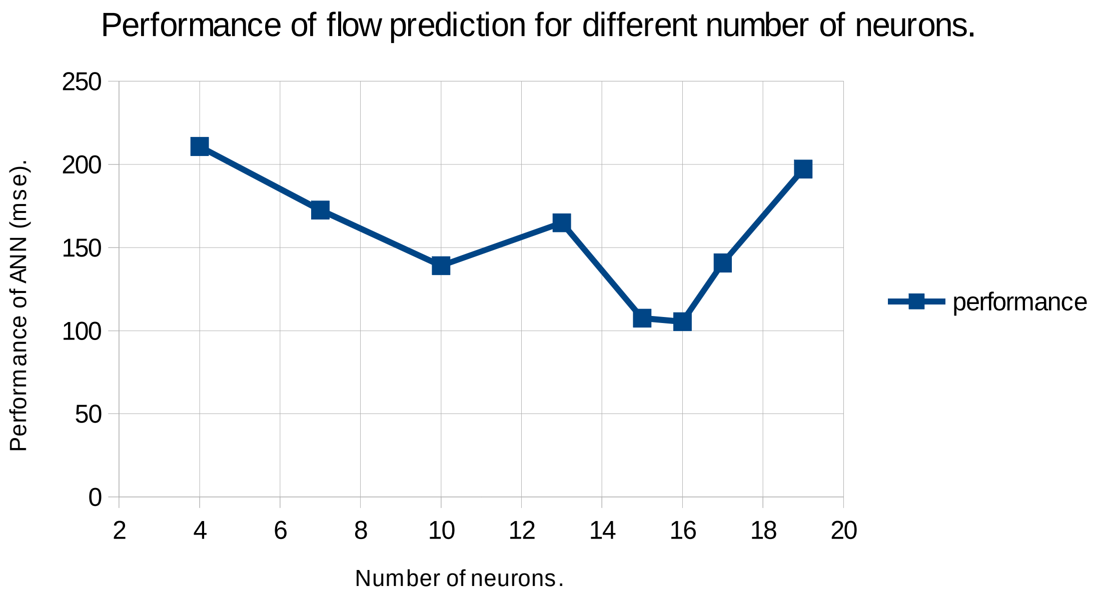


Figure 39.- Performance of ANN for traffic flow prediction using 14 inputs and different number of neurons.

In figure 40 we can observe the structure of the ANN with 14 inputs and 16 neurons as a result of this optimization process. Remember that the number of delays is the double of the number of inputs.

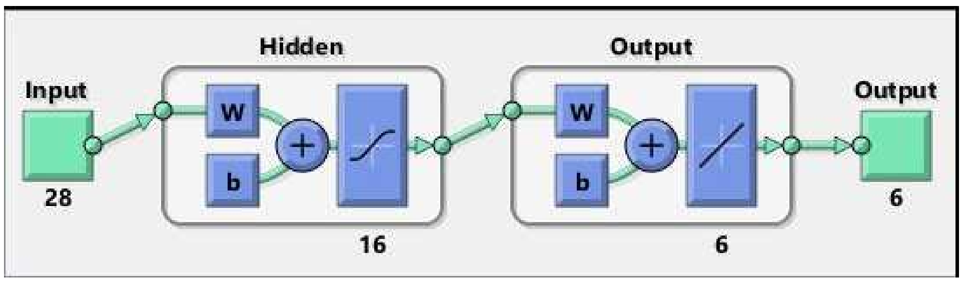


Figure 40.- Structure of the ANN for optimized traffic flow prediction.

In figure 41, we can observe the Root Mean Square Error (RMSE) for the optimized ANN for flow prediction, this value goes from 6.8 to 12.8 vehicles per every 5 minutes.

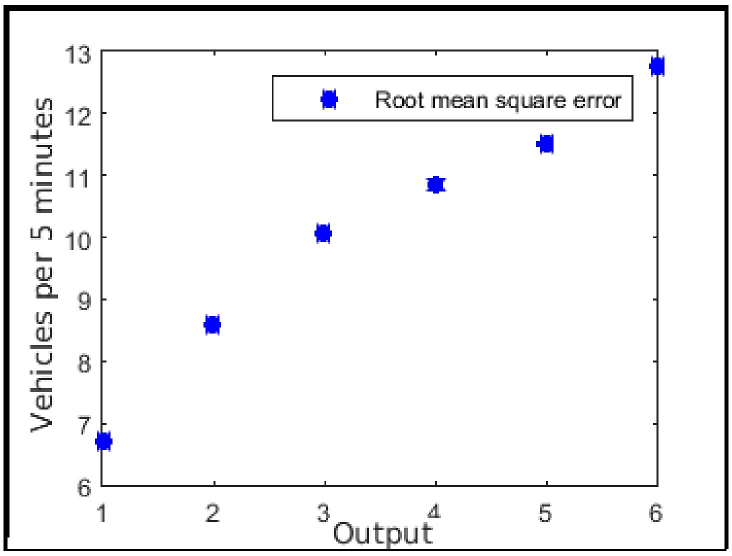


Figure 41.- Root Mean Square Error (RMSE) of every output of the optimized ANN for flow prediction.

In figure 42, we can observe the Mean Absolute Percent Error (MAPE) for the optimized ANN for flow prediction. It goes from 2 to 4 % (98%-96% of good prediction) that compared with the nonoptimized ANN is similar but better than the one in figure 16 with a value of 86%.

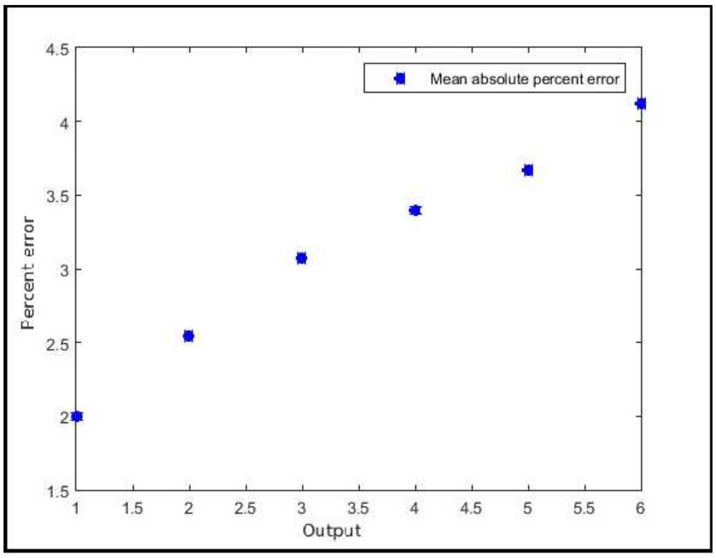


Figure 42.- Mean Absolute Percent Error (MAPE) of every output of the optimized ANN for flow prediction.

In figure 43, we can observe the response for flow for the six outputs. The outputs start from top to right and then bottom left to bottom right. This means that the 5 minutes ahead prediction is in top left corner, and 15 minutes ahead prediction is in top right corner, 20 minutes ahead prediction is in bottom left corner, and 30 minutes ahead prediction is bottom right corner. We can observe how the predicted flow in red dots follows very closely the blue line of the real measures of flow.

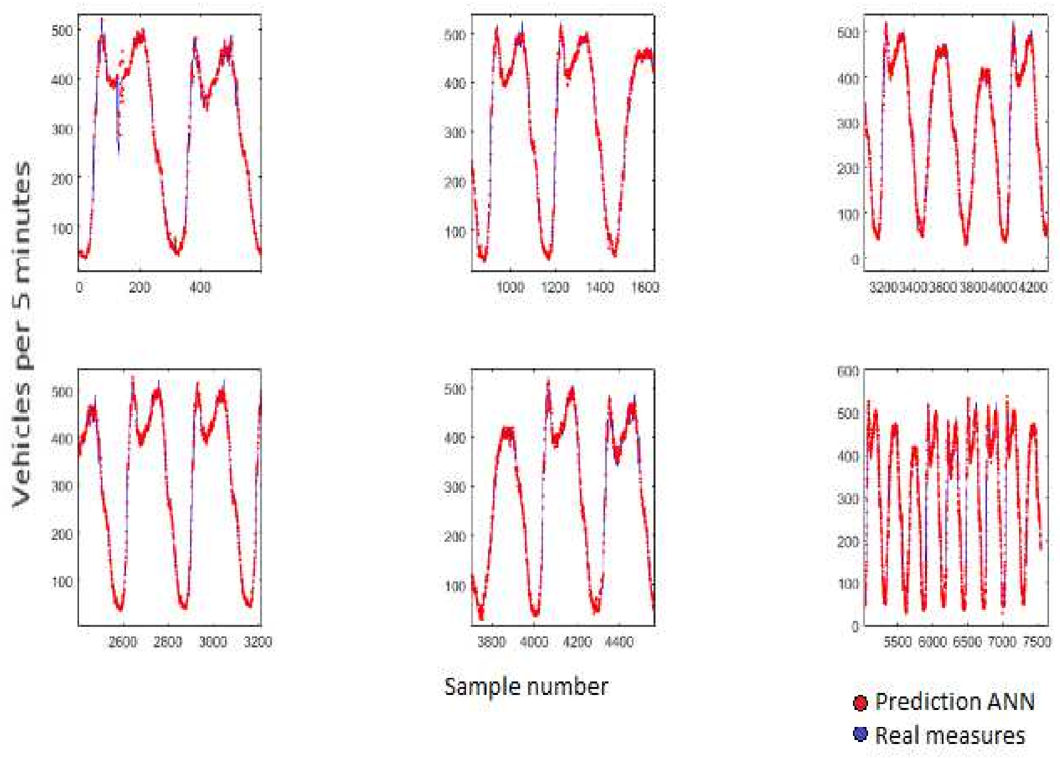


Figure 43.- Response of the six outputs (30 minutes prediction horizon) for traffic flow.

2.8.5. Sensibility analysis.

We are applying several inputs to the feedforward neural network as an increment or decrement, both are applied, and the derivative is taken as an approximation to the ideal derivative as follows:

$$\frac{\triangle Y}{\triangle X}$$

and we are taking the mean of the derivatives and put them into a table. We also calculated the standard deviation and put them into a table. This numbers indicate the derivative of the output *y*, with respect to the entrance *x*, and the output *y. Notice that in the case of y the derivative is taken with respect to the past output for example: y(t+1)...y(t+6), with respect to y(t-0), y(t-1), y(t-2),… .*

Sensibility analysis for speed prediction.

In table 6, we can observe the sensibility analysis results for speed. For the speed case we have eighteen delays for x, and eighteen delays for y. We can observe also that the variation of the output *y, with respect to the input y itself is in the order of magnitude of centesimals, while the variation of the output y, with respect to the input x is in the order of microunits.* This lead us to conclude that the ANN to predict the speed is more sensible to y, the speed previous outputs, than to *x*, the time inputs at which the speed was sensed.

In table 7, we show the standard deviation of the mean derivative shown in table 6.

Sensibility analysis for flow prediction.

In table 8, we can observe the sensibility analysis for flow. For the flow case we have fourteen delays for x, and fourteen delays for y. We can observe also that the variation of the output *y, with respect to the input y itself is in the order of centesimals, while the output y with respect to the x input is in the order of ten thousandth.* We can conclude from this that the output y is more sensible to the output y itself , the flow, than to the input x, the time it was sensed the flow.

In table 9, we show the standard deviation of the mean derivative shown in table 8. Both speed and flow show similar sensibility results.

Table 6.- Mean derivative of speed.

MEAN	y(t+1)	y(t+2)	y(t+3)	y(t+4)	y(t+5)	y(t+6)
x(t-0)	4.71E-006	6.31E-006	6.86E-006	7.09E-006	7.36E-006	7.27E-006
x(t-1)	-6.18E-006	-7.68E-006	-8.42E-006	-9.04E-006	-9.97E-006	-1.03E-005
x(t-2)	-1.27E-006	-1.73E-006	-1.79E-006	-1.36E-006	-7.24E-007	-2.04E-007
x(t-3)	-6.16E-007	-9.94E-007	-8.42E-007	-1.04E-007	9.02E-007	1.73E-006
x(t-4)	6.62E-007	-7.29E-007	-1.50E-006	-1.79E-006	-1.98E-006	-2.11E-006
x(t-5)	2.93E-005	3.12E-005	3.31E-005	3.44E-005	3.49E-005	3.51E-005
x(t-6)	-5.94E-006	-1.23E-006	2.58E-006	4.59E-006	6.27E-006	6.61E-006
x(t-7)	-1.49E-005	-1.64E-005	-1.75E-005	-1.77E-005	-1.69E-005	-1.62E-005
x(t-8)	-3.92E-006	-2.86E-006	-2.27E-006	-1.92E-006	-1.41E-006	-1.08E-006
x(t-9)	7.22E-007	-3.14E-006	-6.16E-006	-7.89E-006	-9.35E-006	-1.03E-005
x(t-10)	3.23E-006	3.49E-006	3.14E-006	2.16E-006	9.79E-007	1.45E-007
x(t-11)	-6.31E-006	-1.49E-005	-2.13E-005	-2.42E-005	-2.62E-005	-2.61E-005
x(t-12)	3.90E-006	6.89E-006	9.34E-006	1.00E-005	9.92E-006	9.56E-006
x(t-13)	-1.81E-006	-6.64E-007	-1.33E-007	-8.29E-008	1.29E-008	-9.18E-008
x(t-14)	1.07E-006	9.30E-007	8.42E-007	1.17E-006	1.65E-006	2.15E-006
x(t-15)	2.73E-006	3.53E-006	4.17E-006	4.50E-006	4.76E-006	4.62E-006
x(t-16)	-1.69E-006	2.05E-007	1.47E-006	1.83E-006	1.95E-006	1.60E-006
x(t-17)	-7.09E-006	-1.07E-005	-1.36E-005	-1.51E-005	-1.61E-005	-1.64E-005
y(t-0)	0.045714377	0.054208131	0.054765421	0.055455112	0.061250969	0.063601826
y(t-1)	-0.029695276	-0.008183018	0.00646343	0.011109109	0.014749509	0.013112375
y(t-2)	0.050834505	0.041430122	0.035734469	0.032449249	0.027808835	0.026062505
y(t-3)	-0.01250654	-0.001265365	0.004771444	0.005735326	0.006754562	0.00618144
y(t-4)	-0.010453971	0.009434618	0.023546601	0.028127849	0.030722227	0.030965253
y(t-5)	-0.000308832	0.002982862	0.002825112	0.00095868	-0.000111172	-0.000459723
y(t-6)	0.011449981	0.005053572	-0.003456142	-0.013333494	-0.02489253	-0.03229326
y(t-7)	-0.01759813	-0.00390316	0.003119062	0.000830992	-0.002786431	-0.007813256
y(t-8)	0.029063838	0.040436184	0.050321075	0.057405289	0.064366165	0.069553888
y(t-9)	0.014183377	0.013501858	0.007308599	0.004267724	0.004840688	0.005437548
y(t-10)	0.012144466	0.012141184	0.011290358	0.011106052	0.012214236	0.013599824
y(t-11)	0.040462588	0.05244601	0.06175272	0.06343251	0.062156332	0.058662429
y(t-12)	0.063366566	0.028311571	0.002602188	-0.012090584	-0.026171515	-0.033319798
y(t-13)	-0.083303397	-0.015590397	0.041446644	0.067872966	0.088604765	0.092064068
y(t-14)	0.107497005	0.096447893	0.089982733	0.08985577	0.090437643	0.090461267
y(t-15)	0.071662644	0.088639196	0.102710328	0.114288272	0.125674654	0.129521718
y(t-16)	-0.181116062	-0.075051517	0.009232406	0.053823726	0.095492483	0.110573438
y(t-17)	0.483312952	0.209069938	0.000531248	-0.089795156	-0.165123398	-0.176917542

Table 7.- Standard deviation of the derivatives for speed.

STD.DEV.	y(t+1)	y(t+2)	y(t+3)	y(t+4)	y(t+5)	y(t+6)
x(t-0)	4.72E-010	8.66E-010	1.17E-009	1.33E-009	1.44E-009	1.49E-009
x(t-1)	2.70E-010	4.53E-010	6.09E-010	7.01E-010	7.68E-010	7.88E-010
x(t-2)	7.28E-011	1.43E-010	1.87E-010	1.91E-010	1.71E-010	1.50E-010
x(t-3)	4.80E-011	6.12E-011	1.82E-010	3.23E-010	5.20E-010	6.62E-010
x(t-4)	5.34E-011	9.56E-011	1.25E-010	1.24E-010	9.94E-011	7.97E-011
x(t-5)	6.53E-010	1.35E-009	1.79E-009	2.04E-009	2.22E-009	2.31E-009
x(t-6)	6.35E-010	1.03E-009	1.35E-009	1.56E-009	1.71E-009	1.79E-009
x(t-7)	1.06E-010	2.15E-010	2.66E-010	2.70E-010	2.48E-010	2.16E-010
x(t-8)	2.58E-010	4.97E-010	6.60E-010	7.12E-010	7.26E-010	7.15E-010
x(t-9)	2.98E-010	1.39E-009	2.23E-009	2.75E-009	3.28E-009	3.58E-009
x(t-10)	3.69E-010	7.99E-010	1.09E-009	1.19E-009	1.23E-009	1.21E-009
x(t-11)	3.28E-011	1.89E-010	3.19E-010	3.74E-010	4.20E-010	4.48E-010
x(t-12)	1.86E-010	4.00E-010	5.51E-010	6.01E-010	6.21E-010	6.26E-010
x(t-13)	1.07E-011	5.10E-012	3.56E-012	1.58E-011	2.73E-011	3.37E-011
x(t-14)	6.88E-011	1.22E-010	1.59E-010	1.71E-010	1.74E-010	1.72E-010
x(t-15)	1.72E-011	1.83E-011	1.87E-011	1.65E-011	1.29E-011	1.08E-011
x(t-16)	6.19E-011	1.58E-010	2.22E-010	2.62E-010	2.97E-010	3.14E-010
x(t-17)	1.92E-010	2.03E-010	2.20E-010	2.51E-010	2.84E-010	3.04E-010
y(t-0)	0.026471575	0.040596861	0.05199749	0.05745001	0.060998396	0.060895546
y(t-1)	0.004986255	0.007751313	0.00976569	0.011230579	0.011700385	0.011914019
y(t-2)	0.000836298	0.00061125	0.000564511	0.000542629	0.000511799	0.000475986
y(t-3)	0.003153009	0.006178944	0.008048515	0.008623837	0.008782183	0.00850894
y(t-4)	0.003476233	0.005795801	0.007155076	0.007439054	0.00732724	0.006368431
y(t-5)	0.003779609	0.006663654	0.0114846	0.014045317	0.015561428	0.015689276
y(t-6)	0.005394488	0.002161641	0.004661493	0.008276513	0.012726717	0.016021016
y(t-7)	0.002661576	0.003848007	0.004661701	0.00507783	0.00558539	0.005805119
y(t-8)	0.013185512	0.007849967	0.005469261	0.005268495	0.005569836	0.006007919
y(t-9)	0.002748468	0.004928873	0.007654621	0.009273941	0.01064283	0.011398127
y(t-10)	0.00326582	0.007021235	0.009895479	0.010328713	0.010010277	0.008698898
y(t-11)	0.007082179	0.005042264	0.003767584	0.003777665	0.004152535	0.004659778
y(t-12)	0.001716667	0.002406875	0.004070359	0.005090843	0.006563124	0.007776819
y(t-13)	0.006109359	0.001452711	0.005537661	0.006694475	0.006950266	0.006278722
y(t-14)	0.004583064	0.005285333	0.006248236	0.006714527	0.006908818	0.006923089
y(t-15)	0.007996392	0.014533009	0.01966962	0.02250909	0.024484144	0.025123863
y(t-16)	0.017313118	0.01470216	0.015019284	0.016453542	0.018187254	0.019337345
y(t-17)	0.037218149	0.034578883	0.045990503	0.052259744	0.057440275	0.056190855

MEAN	y(t+1)	y(t+2)	y(t+3)	y(t+4)	y(t+5)	y(t+6)
x(t-0)	0.0001246	-0.000121501	-0.000355876	-0.000566309	-0.000703007	-0.000747407
x(t-1)	-0.000403826	-0.000117171	-0.000435116	-0.000887982	-0.001092208	-0.0008608
x(t-2)	0.00124551	0.001424929	0.001651955	0.001733142	0.001663804	0.001536916
x(t-3)	-0.00026201	0.000198975	0.000721253	0.00110664	0.001267893	0.001241922
x(t-4)	-0.000324467	-0.0004094	-4.36E-005	0.000561297	0.000967942	0.000833362
x(t-5)	-0.000751494	-0.001186172	-0.001161336	-0.000741947	-0.000231856	3.64E-005
x(t-6)	-0.000257899	-0.000723465	-0.001393865	-0.001920147	-0.001914131	-0.001315604
x(t-7)	0.000480459	0.000177953	-0.000383582	-0.00086342	-0.001061581	-0.001000899
x(t-8)	0.000337232	0.000336288	0.000172037	-0.000110077	-0.000451904	-0.0007492
x(t-9)	-0.000866817	-0.00050945	6.43E-006	0.000205381	-0.000126263	-0.00073772
x(t-10)	-0.00027081	-0.000371505	-0.000260289	-0.000114699	-0.000138977	-0.000418534
x(t-11)	-0.000458707	-0.000538027	-0.000734182	-0.00075829	-0.000623184	-0.000494945
x(t-12)	-0.000286905	-8.40E-005	-0.000113656	-0.00018289	-7.23E-005	0.000279467
x(t-13)	0.001520842	0.001544115	0.001723401	0.00174105	0.001580553	0.001377275
y(t-0)	-0.06198151	-0.087412008	-0.05046305	0.011484197	0.045577091	0.025224359
y(t-1)	-0.013983143	0.000567129	-0.040429004	-0.04819155	0.009597906	0.092980597
y(t-2)	0.062928687	-0.003013909	-0.014740807	-0.032134627	-0.060545838	-0.089382844
y(t-3)	0.0254221	0.101357099	0.071662957	-0.012634579	-0.057567841	-0.000760513
y(t-4)	-0.002128579	0.020708511	0.009291735	-0.018237683	-0.075883896	-0.150881646
y(t-5)	-0.045478588	-0.060247462	0.0002325	0.015256405	-0.047971637	-0.145156573
y(t-6)	0.040992844	0.012855614	-0.018937836	-0.022969775	-0.011878144	-0.011404554
y(t-7)	-0.047446731	-0.063400218	-0.020088861	0.040811512	0.07644766	0.058776966
y(t-8)	0.003474217	-0.019114581	-0.038802659	-0.009638344	0.04852783	0.089236761
y(t-9)	-0.019475703	-0.028944493	-0.066733707	-0.050015675	0.02143031	0.093188897
y(t-10)	0.107529415	0.179333113	0.2095017	0.158638954	0.084704841	0.064471038
y(t-11)	-0.055842749	-0.002310715	0.027620757	-0.010564439	-0.061833824	-0.051092273
y(t-12)	-0.063029744	-0.057516217	0.003557673	0.05736146	0.056142034	0.005862633
y(t-13)	1.028268801	0.942827804	0.853076623	0.846283306	0.899277102	0.93504797

Table 8.- Mean derivatives of flow.

Table 9.- Standard deviation of the derivatives for flow.

STD.DEV.	y(t+1)	y(t+2)	y(t+3)	y(t+4)	y(t+5)	y(t+6)
x(t-0)	1.84E-008	2.34E-007	7.29E-007	1.01E-006	1.12E-006	1.15E-006
x(t-1)	2.01E-007	3.30E-007	7.32E-007	8.51E-007	8.53E-007	9.50E-007
x(t-2)	8.72E-008	6.21E-008	6.83E-008	8.70E-008	1.13E-007	1.40E-007
x(t-3)	1.47E-008	3.54E-008	8.52E-008	1.23E-007	1.49E-007	1.66E-007
x(t-4)	2.32E-007	1.44E-007	2.77E-008	1.27E-007	2.30E-007	2.45E-007
x(t-5)	2.96E-008	1.74E-008	2.39E-008	5.92E-008	1.01E-007	1.52E-007
x(t-6)	2.44E-007	1.97E-007	3.00E-007	6.00E-007	9.42E-007	1.13E-006
x(t-7)	1.95E-008	3.96E-008	6.25E-008	8.27E-008	1.09E-007	1.23E-007
x(t-8)	1.41E-008	2.39E-008	3.12E-008	1.71E-008	2.39E-008	3.82E-008
x(t-9)	1.53E-008	8.79E-008	1.33E-007	1.45E-007	1.74E-007	1.88E-007
x(t-10)	6.90E-009	8.01E-008	2.81E-007	4.88E-007	6.32E-007	6.81E-007
x(t-11)	2.75E-007	3.19E-007	3.09E-007	3.09E-007	3.16E-007	3.17E-007
x(t-12)	3.72E-008	1.31E-007	2.92E-007	4.16E-007	5.39E-007	6.56E-007
x(t-13)	9.13E-009	2.35E-007	4.02E-007	4.58E-007	4.50E-007	4.49E-007
y(t-0)	0.003303986	0.003641691	0.004461523	0.005662536	0.006120329	0.004771089
y(t-1)	0.011827072	0.013275683	0.008421897	0.005486035	0.006734134	0.010267676
y(t-2)	0.00969855	0.016759975	0.027092413	0.028650653	0.025729707	0.022447249
y(t-3)	0.002864698	0.006247678	0.004184112	0.002014414	0.005977437	0.005853214
y(t-4)	0.029496156	0.013912243	0.007466699	0.012750895	0.013747979	0.013882069
y(t-5)	0.006585982	0.006631512	0.004929742	0.003174299	0.008637299	0.016480491
y(t-6)	0.007379323	0.006798028	0.005225379	0.002796242	0.000873704	0.000418609
y(t-7)	0.004653403	0.00300131	0.002152089	0.008260664	0.013293123	0.014915085
y(t-8)	0.001827993	0.002334983	0.004740339	0.003901021	0.001622622	0.000729745
y(t-9)	0.004153808	0.004119727	0.005517736	0.005092731	0.002537116	0.001701865
y(t-10)	0.023677066	0.024490069	0.022097646	0.013130023	0.003634353	0.005723569
y(t-11)	0.00702412	0.008914501	0.011426064	0.009386276	0.005324383	0.003949099
y(t-12)	0.011231886	0.009565813	0.002109051	0.007475595	0.014107752	0.018377005
y(t-13)	0.006134287	0.024200631	0.039382572	0.047062112	0.050115456	0.052329342

3. CONCLUSIONS

3.1. Conclusions

1. The Levenberg-Marquardt algorithm presents better results and performance in the ANN than the backpropagation algorithm. In figure 35, we can observe the Root Mean Square Error (RMSE) for the optimized ANN for speed prediction. It goes from 0.65 – 0.95 mph which compared to figure 8, whose values goes from 2 to 6 mph, it's a decrease that goes from 4 to 6 times. In figure 36, we can observe the the Mean Absolute Percent Error (MAPE) for the optimized ANN for speed prediction. The MAPE goes from 0.57-0.95 % that compared with the values from figure 6b and 7b which are peaks that goes from 30 to 50% is a dramatical reduction in the error of 30 – 50 times less error.

2. There is no need to switch between two different algorithms

3.2. Future work.

Being able to predict spatial impact of accidents and roads congestions. Also it would be interesting to experiment with deep learning techniques and the Levenberg-Marquardt training algorithm in ordet to see how much the performance is improved. We can also recommend the usage of Hadoop to process the unstructured information -text files- obtained in order to do the calculations of speed and flow through ANNs.

BIBLIOGRAPHY.

[1] “IBM big data for the automotive industry”. IBM Software white paper. 2013. Available at: http://www.oesa.org/Doc-Vault/Knowledge-Center/Operational-Performance-Content/IBM-Big-Data-for-Auto-Industry.pdf.

[2] Insurance Institute for Highway Safety. “Crash avoidance features by make and model”. 1996-2014. Available at: http://www.iihs.org/iihs/ratings/crash-avoidance-features

[3] Insurance Institute for Highway Safety. “Crash avoidance features. Topic overview.” 1996-2014. Available at: http://www.iihs.org/iihs/topics/t/crash-avoidance-technologies/topicoverview

[4] Aldana, Karen. “U.S. Department of Transportation Releases Policy on Automated Vehicle Development”. May 30th. 2013. Available at: http://www.nhtsa.gov/About+NHTSA/Press+Releases/U.S.+Department+of+Transportation+Releases+Policy+on+Automated+Vehicle+Development

[5] T Systems. “Big Data in Automotive”. Available at: http://cloud.t-systems.com/whitepaper/14576/dl-big-data-in-automotive.pdf

[6] SAS. “Big Data what it is and why it matters”. Available at: http://www.sas.com/en_us/insights/big-data/what-is-big-data.html

[7] SAS. “Big Data meets big data analytics”. Available at: http://www.sas.com/content/dam/SAS/en_us/doc/whitepaper1/big-data-meets-big-data-analytics-105777.pdf

[8] Pan, Bei; Demiryurek, Ugur; Shahabi, Cyrus. “Utilizing Real World transportation data for accurate traffic prediction”. Integrated media center. University of Southern California. Los Angeles, US. Available: http://infolab.usc.edu/DocsDemos/ICDM2012.pdf.

[9] Fang Xujian, Wu Hao, Seng Dewen and Xu Haitao. School of Software Engineering, Hangzhou Dianzi University, Hangzhou, China Journal of Chemical and Pharmaceutical Research, 2013, 5(12):565-569

[10] Ascent thought leadership from Atos. “White Paper: Real time traffic forecast”. Available at: http://atos.net/content/dam/global/ascent-whitepapers/ascent-whitepaper-real-time-traffic-forecast.pdf

[11] Matlab Help. MathWorks Inc.

APPENDIX A. The FTP client.

This is the FTPClient used to connect to CALTRANS PEMS system to download the metrics files for the sensors deployed in California highways.

```
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.OutputStream;
import java.io.InputStream;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Properties;
import org.apache.commons.net.ftp.FTP;
import org.apache.commons.net.ftp.FTPClient;
import org.apache.commons.net.ftp.FTPFile;
import org.apache.commons.net.ftp.FTPReply;

public class MainFTPClient {
private static PrintWriter fileOut;

//Print server reply to log file
public static void showServerReply(FTPClient ftpClient) {
String[] replies = ftpClient.getReplyStrings();
if (replies != null && replies.length > 0) {
for (String aReply : replies) {
fileOut.println("SERVER: " + aReply);
fileOut.flush();
}
}
}
public static void main(String[] args) {
String server = "";
int port = 21; //FTP port
//If true connect to remote server, else connect to local server
String LOCAL_FTP = "FALSE";
//remote directory from which to download data.
String remoteDirectory = "";
//Local directory to which we'll save the data
String localDir = "";
//Password for the ftp server.
String pass;
//username ftp connection.
String user;
//Instantiate FTP client
FTPClient ftpClient = new FTPClient();
```

```
try {
//If true connect to remote server, else connect to local server
if(LOCAL_FTP.equalsIgnoreCase("TRUE"))
{
server = "";
user = "";
pass = "";
remoteDirectory = "/home/oracle-06/ftp/pems/";
localDir = "/home/oracle-05m/Documents/USVA/testftp/";
}
else
{
server = "";
user = "";
pass = "";
remoteDirectory = "/D7/Data/5min";
localDir = "/home/oracle-05m/Documents/USVA/pems/ExpData/";
}

//Instantiate a file to write log information at the moment of
downloading data
String fileftplog = "FTPLog.txt";
FileWriter fw = new FileWriter (fileftplog);
BufferedWriter bw = new BufferedWriter (fw);
fileOut = new PrintWriter (bw);
//Instantiate current date
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
//Log the date into FTPLog.txt
fileOut.println(dateFormat.format(date));
fileOut.flush();
//connect to server
ftpClient.connect(server, port);
//print server replies.
showServerReply(ftpClient);
int replyCode = ftpClient.getReplyCode();
//Verify server reply is positive to continue downloading, otherwise log
error and disconnect
if (!FTPReply.isPositiveCompletion(replyCode)) {
fileOut.println("Operation failed. Server reply code: " + replyCode);
fileOut.flush();
ftpClient.disconnect();
return;
}
//Log in into server
boolean success = ftpClient.login(user, pass);
showServerReply(ftpClient);
//If not successful, log error and log out; otherwise log successand
continue.
if (!success) {
fileOut.println("Could not login to the server");
fileOut.flush();
ftpClient.logout();
return;
} else {
```

```
fileOut.println("LOGGED IN SERVER");
fileOut.flush();
}
//Set ftp transfer type to binary.
ftpClient.setFileType( FTP.BINARY_FILE_TYPE);
//enter passive mode
ftpClient.enterLocalPassiveMode();
showServerReply(ftpClient);
//get system name
fileOut.println("Remote system is " + ftpClient.getSystemType());
fileOut.flush();
//change current directory
ftpClient.changeWorkingDirectory(remoteDirectory);
showServerReply(ftpClient);
fileOut.println("Current directory is " +
ftpClient.printWorkingDirectory());
fileOut.flush();
//get list of filenames
FTPFile[] ftpFiles = ftpClient.listFiles();
showServerReply(ftpClient);
int i =1;
if (ftpFiles != null && ftpFiles.length > 0) {
//loop thru files
i=1;
for (FTPFile file : ftpFiles) {
//if file is a directory continue with next entry.
if (!file.isFile()) {
continue;
}
fileOut.println("File is " + file.getName());
fileOut.flush();
String remoteFile2 = file.getName();
File downloadFile2 = new File(localDir+remoteFile2);
//instantiate two streams, one for reading and one for writing
OutputStream outputStream2 = new BufferedOutputStream(new
FileOutputStream(downloadFile2));
InputStream inputStream = ftpClient.retrieveFileStream(remoteFile2);
//instantiate a buffer for reading data and writing it gradually to the
output stream
byte[] bytesArray = new byte[4096];
int bytesRead = -1;
while ((bytesRead = inputStream.read(bytesArray)) != -1) {
outputStream2.write(bytesArray, 0, bytesRead);
}
showServerReply(ftpClient);
success = ftpClient.completePendingCommand();
showServerReply(ftpClient);
if (success) {
fileOut.println("File #" + i + " has been downloaded successfully.");
fileOut.flush();
i++;
}
//close streams
outputStream2.close();
inputStream.close();
}
}
```

```
// logout and disconnect from ftp client.
ftpClient.logout();
ftpClient.disconnect();
} catch (IOException ex) {
fileOut.println("Oops! Something wrong happened\n");
fileOut.flush();
ex.printStackTrace();
} finally {
try {
//if an exception occurs and you are connected,
//logout and disconnect from server
if (ftpClient.isConnected()) {
ftpClient.logout();
ftpClient.disconnect();
fileOut.close();
fileOut.flush();
}
} catch (IOException ex) {
ex.printStackTrace();
}
}
}
}
```

Appendix B .- XmlMetrics2DB.

This is the program that inserts into the DB (MySQL) the measurements of the sensors. For our case we used sensor 715898.

```
import java.util.List;
import java.io.File;
import java.io.PrintWriter;
import java.sql.*;

public class pemsLocalDB {

public pemsLocalDB()
{
}
static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
//static final String JDBC_URL = "jdbc:mysql://127.0.0.1/trafic";
static final String JDBC_URL = "jdbc:mysql://localhost/trafic";
static final String USER = "";
static final String PASS = "------";
Connection conn;
Statement stmt;
ResultSet rs;
/*
public void testTraficDB() throws ClassNotFoundException{
String sql_stmt;
try{
System.out.println("Registering driver...");
Class.forName(JDBC_DRIVER);
System.out.println("Connecting to DB...");
conn = DriverManager.getConnection(JDBC_URL, USER, PASS);
stmt= conn.createStatement();
Timestamp tmStmp= Timestamp.valueOf("2015-06-08 19:50:00");
//Timestamp tmStmp= Timestamp.valueOf("20150601173500");
double flow = 65.5;
double speed = 40.3;
double occupancy = 0.25;
for(vdsStation stn : vdsStnList){
//Insert data into the DB
sql_stmt = "INSERT INTO vds_stations " + "VALUES ("+
stn.getId()+",'"+tmStmp.toString()+"',"+stn.getType()+",'"+
stn.getCountyId()+"','"+stn.getCityId()+"','"+stn.getFreewayId()+"','"+st
n.getFreewayDir()+
"',"+stn.getLanes()+","+ flow +","+speed+","+occupancy+")";
System.out.println(sql_stmt);
stmt.executeUpdate(sql_stmt);
}
//Retrieve info from DB.
System.out.println("Creating SELECT statement...");
sql_stmt = "SELECT * from vds_stations";
rs = stmt.executeQuery(sql_stmt);
while(rs.next()){
int id_vds_station = rs.getInt("id_vds_station"); //vds_id
//Date sensed_datetime = rs.getDate("sensed_datetime"); //Date
Timestamp sensed_datetime = rs.getTimestamp("sensed_datetime"); //Date
int type = rs.getInt("type"); //type
String county_id = rs.getString("county_id");//county_id
String city_id = rs.getString("city_id");//city_id
String freeway_id = rs.getString("freeway_id");//freeway_id
String freeway_dir = rs.getString("freeway_dir");//freeway_dir
int lanes = rs.getInt("lanes");//lanes
flow = rs.getDouble("flow");//flow
speed = rs.getDouble("speed");//speed
occupancy = rs.getDouble("occupancy");//occupancy
System.out.println("------------------------------------");
System.out.println("id_vds_station: "+ id_vds_station);
System.out.println("sensed_datetime: " + sensed_datetime.toString());
System.out.println("type: " + type);
System.out.println("county_id: "+ county_id);
System.out.println("city_id: " + city_id);
System.out.println("freeway_id: "+freeway_id);
System.out.println("freeway_dir: "+freeway_dir);
System.out.println("lanes: " + lanes);
System.out.println("flow: " + flow);
System.out.println("speed: "+ speed);
System.out.println("occupancy: "+ occupancy);
System.out.println("------------------------------------");
}
}catch(Exception e)
{
e.printStackTrace();
}finally{
//finally block used to close resources
try{
if(stmt!=null)
stmt.close();
}catch(SQLException se2){
}// nothing we can do
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
se.printStackTrace();
}//end finally try
}//end try
System.out.println("Goodbye...");
}
*/
//registry and connect to database
public void regAndConn2DB() throws ClassNotFoundException{

try{
System.out.println("Registering driver...");
Class.forName(JDBC_DRIVER);
System.out.println("Connecting to DB...");
conn = DriverManager.getConnection(JDBC_URL, USER, PASS);
stmt = conn.createStatement();
}catch(Exception e)
{
e.toString();
e.printStackTrace();
System.exit(-1);
```

```
}
}
//insert station and metrics (realtime measurements) into database (only
one record)
public void insertStationAndMetrics(vdsStation stn,vdsStnMetrics vdsStnM,
String date, double time) {
//Timestamp tmStmp= Timestamp.valueOf("2015-06-08 19:50:00");
String sql_stmt;
try{
//Get flow, speed, and occupancy
double flow = vdsStnM.getFlow();
double speed = vdsStnM.getSpeed();
double occupancy = vdsStnM.getOccupancy();
//create string that represents the SQL statement
sql_stmt = "INSERT INTO vds_stations " + "VALUES
("+stn.getDistrict()+","+
stn.getId()+",'"+date+"',"+time+","+stn.getType()+",'"+
stn.getCountyId()+"','"+stn.getCityId()+"','"+stn.getFreewayId()+"','"+st
n.getFreewayDir()+
"',"+stn.getLanes()+","+ flow
+","+speed+","+occupancy+","+stn.getLatitude()+","+
stn.getLongitude()+")";
//System.out.println(sql_stmt);
//Execute SQL statement
stmt.executeUpdate(sql_stmt);
}catch(Exception e)
{
e.printStackTrace();
}
}
//append a record of metrics to a csv file
public void convert2CSV(vdsStation stn,vdsStnMetrics vdsStnM, String
date, double time, PrintWriter csvFile)
{
String sql_stmt;
try{
//Get flow, speed, and occupancy
double flow = vdsStnM.getFlow();
double speed = vdsStnM.getSpeed();
double occupancy = vdsStnM.getOccupancy();
//create CSV row.
sql_stmt = stn.getDistrict()+","+
stn.getId()+",'"+date+"',"+time+","+stn.getType()+",'"+
stn.getCountyId()+"','"+stn.getCityId()+"','"+stn.getFreewayId()+"','"+st
n.getFreewayDir()+
"',"+stn.getLanes()+","+ flow
+","+speed+","+occupancy+","+stn.getLatitude()+","+
stn.getLongitude();
//System.out.println(sql_stmt);
//Write into csv file
csvFile.println(sql_stmt);
csvFile.flush();
//}
}catch(Exception e)
{
e.printStackTrace();
}
```

```
}
public void disconnect(){
//finally block used to close resources
try{
if(stmt!=null)
stmt.close();
}catch(SQLException se2){
}// nothing we can do
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
se.printStackTrace();
}//end finally try
}
}
```

```java
import java.util.ArrayList;
import java.util.List;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;


public class VdsSaxHandler extends DefaultHandler {
private List<vdsStation> vdsStnList;
private vdsStation vdsStn;
//return vdsStationList.
public List<vdsStation> getVdsStationList(){
return vdsStnList;
}

boolean district = false;
boolean detectorStations = false;
boolean vds = false;
final String venturaLADistrict = "7";
private long nullHitsId=0;
private long nullHitsName=0;
private long nullHitsTypeStr=0;
private long nullHitsCounty=0;
private long nullHitsCity=0;
private long nullHitsFid=0;
private long nullHitsFd=0;
private long nullHitsLanes=0;
private long nullHitsLat=0;
private long nullHitsLong=0;
@Override
public void startElement(String uri, String localName, String qName,
Attributes attributes)
throws SAXException {
if (qName.equalsIgnoreCase("District")) {
//verify if it is district 7 of Los Angeles
String attDistrict = attributes.getValue("id");
if(attDistrict.equalsIgnoreCase("7"))
{
district= true;
System.out.println("District 7 opened");
}
} else if (qName.equalsIgnoreCase("detector_stations")) {
//set detector stations true if the xml tag is "detector_stations"
detectorStations = true;
System.out.println("detector_ stations opened");
} else if (qName.equalsIgnoreCase("vds")) {
//set vds member variable to true if xml tag is "vds"
vds = true;
if(district == true && detectorStations == true){
//instantiate new vdsStation
vdsStn = new vdsStation();
//if it is the first time the vdsStnLIst is empty.
if(vdsStnList == null)
vdsStnList = new ArrayList<>();
long id;
```

```
String name, typeStr, county_id, city_id, freeway_id, freeway_dir;
int type, lanes;
double latitude, longitude;
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("id").equalsIgnoreCase("")){
id= -1;
nullHitsId++;
}else{
id = Long.parseLong(attributes.getValue("id"), 10) ;
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("name").equalsIgnoreCase("")){
name= "";
nullHitsName++;
}else{
name = attributes.getValue("name");
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("type").equalsIgnoreCase("")){
typeStr= "";
nullHitsTypeStr++;
}else{
typeStr = attributes.getValue("type");
}
//get freeway type
type = getFreewayType(typeStr);
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("county_id").equalsIgnoreCase("")){
county_id= "";
nullHitsCounty++;
}else{
county_id = attributes.getValue("county_id");
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("city_id").equalsIgnoreCase("")){
city_id= "";
nullHitsCity++;
}else{
city_id = attributes.getValue("city_id");
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("freeway_id").equalsIgnoreCase("")){
freeway_id= "";
nullHitsFid++;
}else{
freeway_id = attributes.getValue("freeway_id");
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("freeway_dir").equalsIgnoreCase("")){
freeway_dir = "";
nullHitsFd++;
}else{
freeway_dir = attributes.getValue("freeway_dir");
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("lanes").equalsIgnoreCase("")){
lanes = 1;
nullHitsLanes++;
}else{
lanes = Integer.parseInt(attributes.getValue("lanes"),10);
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("latitude").equalsIgnoreCase("")){
latitude = 0.0;
nullHitsLat++;
}else{
latitude = Double.parseDouble(attributes.getValue("latitude"));
}
//verify for nullHits, else parse the attribute and transform it to
approppriate type
if(attributes.getValue("longitude").equalsIgnoreCase("")){
longitude = 0.0;
nullHitsLong++;
}else{
longitude = Double.parseDouble(attributes.getValue("longitude"));
}
//store all information into vdsStation
vdsStn.setDistrict(Integer.parseInt(venturaLADistrict,10));
vdsStn.setId(id);
vdsStn.setName(name);
vdsStn.setType(type);
vdsStn.setCountyId(county_id);
vdsStn.setCityId(city_id);
vdsStn.setFreewayId(freeway_id);
vdsStn.setFreewayDir(freeway_dir);
vdsStn.setLanes(lanes);
vdsStn.setLatitude(latitude);
vdsStn.setLongitude(longitude);

}
}
}
//return an integer representing the freeway number.
private int getFreewayType(String typeStr) {
int typeFW=0;
// TODO Auto-generated method stub
if(typeStr.equalsIgnoreCase("CD"))
typeFW = 1;
else if(typeStr.equalsIgnoreCase("CH"))
typeFW = 2;
else if(typeStr.equalsIgnoreCase("FF"))
typeFW = 3;
else if(typeStr.equalsIgnoreCase("FR"))
typeFW = 4;
else if(typeStr.equalsIgnoreCase("HV"))
```

```java
typeFW = 5;
else if(typeStr.equalsIgnoreCase("ML"))
typeFW = 6;
else if(typeStr.equalsIgnoreCase("OR"))
typeFW = 7;
return typeFW;
}
@Override
public void endElement(String uri, String localName, String qName) throws
SAXException {
if (qName.equalsIgnoreCase("vds")) {
if(district){
//add vds element if we find the closing brace of an xml tag "vds"
vdsStnList.add(vdsStn);
//set vds to false because we found a closing tag for vds
vds = false;
}
}
else if(qName.equalsIgnoreCase("detector_stations")){
//set detectorStations to false because we found a closing tag
detectorStations = false;
}else if(qName.equalsIgnoreCase("District")){
//set district to false because we found a closing tag
district = false;
System.out.println("nullHits");
System.out.println("nullHitsId: "+nullHitsId);
System.out.println("nullHitsName: "+nullHitsName);
System.out.println("nullHitsTypeStr: "+nullHitsTypeStr);
System.out.println("nullHitsCity: "+nullHitsCity);
System.out.println("nullHitsCounty: "+nullHitsCounty);
System.out.println("nullHitsFid: "+nullHitsFid);
System.out.println("nullHitsFd: "+nullHitsFd);
System.out.println("nullHitsLanes: "+nullHitsLanes);
System.out.println("nullHitsLat: "+nullHitsLat);
System.out.println("nullHitsLong: "+nullHitsLong);
}
}
@Override
public void characters(char ch[], int start, int length) throws
SAXException {
//do nothing.
}
}
```

```
//Class that represents a vds station as they are configured in the
vds_config.xml file
public class vdsStation implements Comparable<vdsStation> {
//vdsStation constructor
public vdsStation(){
district = -1;
id =-1;
name = "";
type = 3;
county_id = "";
city_id = "";
freeway_id = "";
freeway_dir = "";
lanes = 1;
latitude=0.0;
longitude=0.0;
}
//private members.
private int district;
private long id;
private String name;
private int type;
private String county_id;
private String city_id;
private String freeway_id;
private String freeway_dir;
private int lanes;
private double latitude;
private double longitude;
//public setter and getter methods.
public void setDistrict(int districtp){this.district = districtp;}
public int getDistrict(){return district;}
public void setId(long vdsid){this.id = vdsid;}
public long getId(){return id;}
public void setName(String namep){this.name = namep;}
public String getName(){return name;}
public void setType(int typep){this.type = typep;}
public int getType(){return type;}
public void setCountyId(String countyIdp){this.county_id = countyIdp;}
public String getCountyId(){return county_id;}
public void setCityId(String cityIdp){this.city_id = cityIdp;}
public String getCityId(){return city_id;}
public void setFreewayId(String freewayIdp){this.freeway_id =
freewayIdp;}
public String getFreewayId(){return freeway_id;}
public void setFreewayDir(String freewayDirp){this.freeway_dir =
freewayDirp;}
public String getFreewayDir(){return freeway_dir;}
public void setLanes(int lanesp){this.lanes = lanesp;}
public int getLanes(){return lanes;}
public void setLatitude(double lat){this.latitude= lat;}
public double getLatitude(){return this.latitude;}
public void setLongitude(double longitude){this.longitude=longitude;}
public double getLongitude(){return this.longitude;}
//function to compare two vds stations and sort them.
public int compareTo(vdsStation o) {
return o.getId() <= this.getId() ? 1: -1;
```

```java
}
//Stringify the object vdsStation for logging purposes.
@Override
public String toString() {
return "vdsStation::"+"district="+this.district+"::ID="+this.id+"::Name="
+ this.name
+"::Type="+type+"::CountyId="+county_id+"::CityID="+city_id
+"::FreewayId="+freeway_id+"::FreewayDir="+freeway_dir
+"::Lanes="+lanes+ "::Latitude= "+latitude+"::Longitude= "+longitude;
}
}
```

```
//Class that represents the metric (real time measurements obtained from
caltrans).
public class vdsStnMetrics implements Comparable<vdsStnMetrics> {
//constructor.
public vdsStnMetrics()
{
vds_Id = -1;
speed = 0.0;
flow = 0.0;
occupancy = 0.0;
}
//private member methods
private long vds_Id;
private double speed;
private double flow;
private double occupancy;
//public setter/getter methods.
public void setVdsId(long id){vds_Id = id;}
public long getVdsId(){return vds_Id;}
public void setSpeed(double speed){this.speed = speed;}
public double getSpeed(){return this.speed;}
public void setFlow(double flow){this.flow = flow;}
public double getFlow(){return this.flow;}
public void setOccupancy(double occupancy){this.occupancy = occupancy;}
public double getOccupancy(){return this.occupancy;}
public int compareTo(vdsStnMetrics o) {
return o.getVdsId() <= this.getVdsId() ? 1: -1;
}
@Override
public String toString(){
return "::VdsId= "+ this.getVdsId() + " ::flow= "+ this.getFlow() + "
::occupancy= "+
this.getOccupancy()+ " ::speed= "+ this.getSpeed();
}
}
```

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.io.IOException;

import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.xml.sax.SAXException;


public class VdsXmlSaxParserImpl {
private static List<vdsStnMetrics> vdsStnMList;
private static List<vdsStation> vdsStnList;
private static pemsLocalDB traficDB=null;
private static long row_ctr=0;
private static PrintWriter fileOut;
private static PrintWriter csvFOut;
public static void main(String [] args){
SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
try {
//Instantiate a saxParser
SAXParser saxParser = saxParserFactory.newSAXParser();
//Instantiate a vdsSaxHandler
VdsSaxHandler handler = new VdsSaxHandler();
//File vdsXmlFile = new File("vds_config_0.xml");
File vdsXmlFile = new File("vds_config.xml");
//parse xml file
saxParser.parse(vdsXmlFile, handler);

//get vdsStationList.
vdsStnList = handler.getVdsStationList();
//sort vdsStationsLists.
Collections.sort(vdsStnList);

//Instantiate file for logging purposes.
//String xmlMetrics2dblog =
"/home/ulisesv/Documents/USVA/ITESO/Thesis/XMLMetrics2DB_Log.txt";
String xmlMetrics2dblog = "/home/oracle-
05m/Documents/USVA/pems/Executables/XML2DBMetrics/XMLMetrics2DB_Log.txt";
FileWriter fw = new FileWriter (xmlMetrics2dblog);
BufferedWriter bw = new BufferedWriter (fw);
fileOut = new PrintWriter (bw);
//Instantiate a file for creating a csv of the metrics.
```

```
String csvF = "/home/oracle-
05m/Documents/USVA/pems/Executables/XML2DBMetrics/vds_stations.csv";
FileWriter fwcsvF = new FileWriter (csvF);
BufferedWriter bwcsvF = new BufferedWriter (fwcsvF);
csvFOut = new PrintWriter (bwcsvF);
/*
for(vdsStation vdsStn : vdsStnList)
System.out.println(vdsStn);
pemsLocalDB pldb = new pemsLocalDB(vdsStnList);
pldb.testTraficDB();
*/
//final File folder = new
File("/home/ulisesv/Documents/USVA/ITESO/Thesis/pems/D7");
//final File folder = new File("/home/oracle-
05m/Documents/USVA/pems/Test_preDB2/ExpData");
final File folder = new File("/home/oracle-
05m/Documents/USVA/pems/ExpData");
listFilesForFolder(folder);
System.out.println("!!!!!!!!!!! DONE !!!!!!!!!!");
fileOut.println("!!!!!!!!!!! DONE !!!!!!!!!!");
fileOut.flush();
}catch (Exception e)
{
e.printStackTrace();
fileOut.close();
}/*catch (/*ParserConfigurationException | SAXException | IOException e)
{
e.printStackTrace();
} catch (ClassNotFoundException e) {
// TODO Auto-generated catch block
System.out.println("ClassNotFoundException...");
e.printStackTrace();
}*/
fileOut.flush();
fileOut.close();
}
private static void parseMetricsFiles(String fileName) {
// TODO Auto-generated method stub
//while not eof
try{
//if first time, instantiate a new array for vdsStnMList
if(vdsStnMList== null)
vdsStnMList = new ArrayList<>();
BufferedReader br = new BufferedReader(new FileReader(fileName));
String line;
long ctr =0;
while ((line = br.readLine()) != null) {
// process the line.
ctr++;
//Skip header
if(ctr == 1)
{
//System.out.println("Line 1");
continue;
}
// split string to obtain values of vdsStnMetrics.
String [] metricsArray = line.split(",");
```

```
vdsStnMetrics vdsStnM = new vdsStnMetrics();
double flow=0.0, occupancy=0.0, speed=0.0;
//if null initialize to zero, otherwise parse the value as double.
if(metricsArray[1].equalsIgnoreCase(""))
{
flow = 0.0;
}
else
{
flow = Double.parseDouble(metricsArray[1]);
}
//if null initialize to zero, otherwise parse the value as double.
if(metricsArray[2].equalsIgnoreCase(""))
{
occupancy = 0.0;
}
else
{
occupancy = Double.parseDouble(metricsArray[2]);
}
//if null initialize to zero, otherwise parse the value as double.
if(metricsArray[3].equalsIgnoreCase(""))
{
speed = 0.0;
}
else
{
speed = Double.parseDouble(metricsArray[3]);
}
//Store metrics into vdsStnM
vdsStnM.setVdsId(Long.parseLong(metricsArray[0], 10));
vdsStnM.setFlow(flow);
vdsStnM.setOccupancy(occupancy);
vdsStnM.setSpeed(speed);
//Add vdsStnM to the list.
vdsStnMList.add(vdsStnM);
//System.out.println(line);
}
// * System.out.println(ctr);
}catch (Exception e)
{
e.printStackTrace();
}
}
public static void listFilesForFolder(final File folder) {
try{
//connect to DB
if(traficDB == null)
{
traficDB = new pemsLocalDB();
traficDB.regAndConn2DB();
row_ctr=0;
}
//Sort files in directory "ExpData"
File files[] = folder.listFiles();
Arrays.sort( files, new Comparator<File>(){
public int compare(final File o1, final File o2) {
```

```
return new Long(((File)o1).lastModified()).compareTo
(new Long(((File) o2).lastModified()));
}
});
boolean nextFile= false;
for (final File fileEntry : files)
{
//if file entry is directory recurse into it.
//This was because originally the metrics files were going to reside in
different
//directories, but at the end they reside inside the same directory
if (fileEntry.isDirectory()) {
listFilesForFolder(fileEntry);
} else {
row_ctr=0;
nextFile= false;
//String fileName = new String(fileEntry.getAbsolutePath().getName());
//Get absolute path of file
String fileNameAbs = new String(fileEntry.getAbsolutePath());
//System.out.println(fileNameAbs);
//fileOut.println(fileNameAbs);

//Timestamp tmstmp =
Timestamp.valueOf(getTimestampFromFileName(fileEntry.getName()));
//get time of file as double
double time =
Double.parseDouble(getTimeFromFileName(fileEntry.getName()));
//get Date from filename and store it as string.
String date= getDateFromFileName(fileEntry.getName());
// * System.out.println(time);
// * System.out.println(date);
if(vdsStnMList != null && !vdsStnMList.isEmpty())
vdsStnMList.clear();
//parse metrics file.
parseMetricsFiles(fileNameAbs);
Collections.sort(vdsStnMList);
// * System.out.println(vdsStnMList.size());
try
{
int bin_index=0;
int min =0;
int max =0;
//for(vdsStnMetrics vdsStnM : vdsStnMList)
//{
//Do binary search of a vdsStn for id 715898
for(int i = 0; i < vdsStnMList.size();++i)
{
min = 0;
max = vdsStnMList.size()-1;
if(vdsStnMList.size()%2==0)
{
bin_index = vdsStnMList.size()/2;
}
else
{
bin_index = (vdsStnMList.size()-1)/2;
}
```

```
while(true)
{
if (vdsStnMList.get(i).getVdsId() < vdsStnList.get(bin_index).getId())
{
//min = se queda igual
max = bin_index-1;
bin_index = min + (max-min)/2;
}
else if (vdsStnMList.get(i).getVdsId() >
vdsStnList.get(bin_index).getId())
{
min = bin_index+1;
//max = se queda igual
bin_index = min + (max-min)/2;
}
else if ( vdsStnMList.get(i).getVdsId() ==
vdsStnList.get(bin_index).getId() )
{
//insertInto database.
//System.out.print("*");
//traficDB.insertStationAndMetrics(vdsStnList.get(bin_index),
vdsStnMList.get(i), date, time);
if(vdsStnMList.get(i).getVdsId() == 715898)
{
//insert record into database
traficDB.insertStationAndMetrics(vdsStnList.get(bin_index),
vdsStnMList.get(i), date, time);
//create csv file from station metrics
//traficDB.convert2CSV(vdsStnList.get(bin_index), vdsStnMList.get(i),
date, time,csvFOut);
row_ctr++;
//print progress information.
if((row_ctr%100) == 0)
System.out.println("*");
if((row_ctr%80000)==0)
System.out.println("");
nextFile= true;
break;
}
}
else if( bin_index <= min || bin_index >= max)
{
break;
}
}
if(nextFile)
break;
}

/*
//System.out.println(vdsStnM);
for(vdsStation vdsStn: vdsStnList)
{
if(vdsStnM.getVdsId() == vdsStn.getId())
{
//insert
traficDB.insertStationAndMetrics(vdsStn, vdsStnM, date, time);
if((row_ctr %100)== 0)
System.out.print("*");
if((row_ctr % 800)==0)
System.out.println("");
row_ctr++;
}
}*/
//}

fileOut.println(""+row_ctr+ " "+ fileNameAbs);
fileOut.flush();
}catch(Exception e){
//if exception occurs close everything
e.printStackTrace(fileOut);
fileOut.flush();
fileOut.close();
csvFOut.flush();
csvFOut.close();
}

// * System.out.println("");
}
}
csvFOut.flush();
csvFOut.close();
}catch(Exception e)
{
//if exception occurs close connection to DB and files.
e.printStackTrace(fileOut);
traficDB.disconnect();
System.out.println(row_ctr);
fileOut.flush();
fileOut.close();
csvFOut.flush();
csvFOut.close();
}finally
{
}
}
//return a string that represent the time the metric o file was created.
private static String getTimeFromFileName(String fileName) {
//5minagg_20150601172000.txt
/*
//8-12 year
String year = fileName.substring(8, 12);
//12-14 month
String month = fileName.substring(12, 14);
//14-16 day
String day = fileName.substring(14, 16);
*/
//16-18 hour
String hour = fileName.substring(16, 18);
//18-20 minute
String minute = fileName.substring(18, 20);
//20 second
String second = fileName.substring(20, 22);
```

```
//String strTmstmp = year +"-"+month+"-"+day+" "+
hour+":"+minute+":"+second;
String strTmstmp = hour+minute+second;
return strTmstmp;
}
//return a string representing the date the file or metric was created.
private static String getDateFromFileName(String fileName) {
//5minagg_20150601172000.txt
//8-12 year
String year = fileName.substring(8, 12);
//12-14 month
String month = fileName.substring(12, 14);
//14-16 day
String day = fileName.substring(14, 16);

String strDate = year +"-"+month+"-"+day;
return strDate;
}
}
```

Appendix C .- Matlab script: nn_ts_3m_2d_traffic.m

This script is the fiorst approach to predict traffic flow using Levenberg-Marquardt learning algorithm for ANN. Additionally, it uses a NARX ANN.

```
%% Clean all work environment
%Example using time series and ANN.
clear all;
close all;
clc;
%% Create ANN
% Here is how to design a neural network that predicts the target series
% from past values of inputs and targets. See narxnet for more details.
load res_vds_715898.mat;
%T=res(:,8)';
T=res_vds(:,11)';
%X=res(:,5)';
X=res_vds(:,4)';
%T={cell2mat(res(:,4))+cell2mat(res(:,5))/100}';
%[X,T] = ph_dataset;
% Split the data into test and training sets. nporcen for training and
% (1-nporcen) % for testing the ANN.
%nporcen=0.9; %
%nporcen=0.67; %
nporcen=0.7;
ndat=round(size(T,2)*nporcen);
Ttrain=T(:,1:ndat);
Xtrain=X(:,1:ndat);
Ttest=T(:,ndat:end);
Xtest=X(:,ndat:end);
% Creation of the ANN with delays and external input and the output Y
net = narxnet(1:20,1:20,10); %net = narxnet(1:20,1:20,10,{},'trainlm');
[Xs,Xi,Ai,Ts] = preparets(net,Xtrain,{},Ttrain);
net = train(net,Xs,Ts,Xi,Ai);
view(net);
Y = net(Xs,Xi,Ai);
figure(1);
plotresponse(Ts,Y);
% Simulation of the ANN with testing data
[Xs,Xi,Ai,Ts] = preparets(net,Xtest,{},Ttest);
Ytest = net(Xs,Xi,Ai);
%% create figures with testing data.
figure(2);
plotresponse(Ts,Ytest);
figure(3);
plot(1:size(Ttest,2),cell2mat(Ttest),'b-
',21:size(Ttest,2),cell2mat(Ytest),'r.','LineWidth',2);
xlabel('Samples every 5 minutes');
ylabel('Vehicular Flow (#vehicules by 5 minutes)');
legend('Traffic','Neural Predictor');
%% Simulation of the NARX network with closed loop
netc=closeloop(net);
view(netc);
```

```
%%
[Xs,Xi,Ai,Ts] = preparets(netc,Xtest,{},Ttest);
ycl = netc(Xs,Xi,Ai);
figure(4);
plot(1:size(Ttest,2),cell2mat(Ttest),'b-
',21:size(Ttest,2),cell2mat(ycl),'r.','LineWidth',2);
xlabel('Samples every 5 minutes');
ylabel('Vehicular Flow (#vehicules by 5 minutes)');
legend('Traffic','Neural Predictor');
%% Simulation of NARX ANN to predict the next output
netp = removedelay(net);
view(netp)
%%
[Xs,Xi,Ai,Ts] = preparets(netp,Xtest,{},Ttest);
ypredic = netp(Xs,Xi,Ai);
figure(5);
plot(1:size(Ttest,2),cell2mat(Ttest),'b-
',21:size(Ttest,2)+1,cell2mat(ypredic),'r.','LineWidth',2);
xlabel('Samples every 5 minutes');
ylabel('Vehicular Flow (#vehicules by 5 minutes)');
legend('Traffic','Neural Predictor');
%%
netcp = removedelay(netc);
view(netcp)
%%
[Xs,Xi,Ai,Ts] = preparets(netcp,Xtest,{},Ttest);
ypredic_netcp = netcp(Xs,Xi,Ai);
figure(6);
plot(1:size(Ttest,2),cell2mat(Ttest),'b-
',21:size(Ttest,2),cell2mat(ypredic_netcp),'r.','LineWidth',2);
xlabel('Samples every 5 minutes');
ylabel('Vehicular Flow (#vehicules by 5 minutes)');
legend('Traffic','Neural Predictor');
```

Appendix D.- Matlab script: nn_ts_3m_2d_traf_ff_speed.m

This script is for predicting the speed of the traffic using feedforward neural network.

```
%% Clean all work environment
% traffic prediction using feedforward ANN
clear all;
close all;
clc;
%% Create ANN
% Here is how to design a neural network that predicts the target series
% from past values of inputs and targets. See narxnet for more details.
load res_vds_715898.mat;
%T=res(:,8)';
T=res_vds(:,12)';
%X=res(:,5)';
X=res_vds(:,4)';
%T={cell2mat(res(:,4))+cell2mat(res(:,5))/100}';
%[X,T] = ph_dataset;
%input number
i=1;
%number of delays for input x, and output y.
ninputs = 18;
%number of outputs of the ANN.
noutputs = 6;
while true
limInf_XT_ff = i; %infierior limit of Xtest
limSup_XT_ff = i + ninputs - 1; %superior limit of XTest
limInf_O_ff = i + ninputs; % inferior limit of Output vector (real time
measurements)
limSup_O_ff = i + noutputs + ninputs - 1; % superior limit of output
vector (real time measurements)
%if superior limit of output vector reached the end of the vector, we
have finished
%arranging the matrix for feedforwardnet
if limSup_O_ff > size(T,2)
break;
end
%arrange the vector Xtest with inputs from X, and the outputs Y from T
vector
XT_ff(i,:) = [X(1,limInf_XT_ff:limSup_XT_ff),
T(1,limInf_XT_ff:limSup_XT_ff)];
%arrange the vector Output, with values from T.
O_ff(i,:) = T(1,limInf_O_ff:limSup_O_ff);
%increase the iterator one position.
i=i+1;
end
%%
%net = feedforwardnet(13);
net = feedforwardnet(13);
X_ff_t = XT_ff'; %2X25216 X_ff_t
O_ff_t = O_ff'; %1X25216 T_ff_t
X_ff_t=cell2mat(X_ff_t); % change data type.
O_ff_t=cell2mat(O_ff_t); % change data type.
```

```
% round down to the nearest integer to split between train and test data.
nporcen=0.7;
ndat=round(size(T,2)*nporcen);
% create x vector for training the ANN
X_ff_t_train = X_ff_t(:,1:ndat);
% create y vector (output) for training the ANN
O_ff_t_train = O_ff_t(:,1:ndat);
%increase the offset by one.
ndat_1 = ndat+1;
% create x vector for testing the ANN
X_ff_t_test = X_ff_t(:,ndat_1:end);
% create y vector (output) for testing the ANN
O_ff_t_test = O_ff_t(:,ndat_1:end);
%%
% Train the feedforward ANN
tic
net = train(net,X_ff_t_train,O_ff_t_train);
toc
%view the structure of the ANN
view(net)
%%
% Obtain the output of the feedforward ANN but with testing data set.
tic
y = net(X_ff_t_test);
toc
%calculate the mean square error of the feedforward ANN.
perf = perform(net,y,O_ff_t_test)
%% Calculate RMSE
N = size(O_ff_t_test,2);
diff = O_ff_t_test - y;
diff_sqd = (diff).^2;
sum_diff_sqd = sum(diff_sqd,2);
sum_diff_sqd_N = sum_diff_sqd / N;
RMSE = sqrt(sum_diff_sqd_N);
figure(1);
plot(1:size(RMSE,1),RMSE','b*','LineWidth',2);
xlabel('Output');
ylabel('Vehicles speed (mph)');
legend('Root mean square error');
%% Calculate MAPE
diff_abs=abs(diff);
diff_abs_div=diff_abs./O_ff_t_test;
diff_abs_div_sum = sum(diff_abs_div,2);
MAPE = diff_abs_div_sum*100/N;
figure(2);
plot(1:size(MAPE,1),MAPE','b*','LineWidth',2);
xlabel('Output');
ylabel('Percent error');
legend('Mean absolute percent error');
%% Plotresponse section
subplot(2,3,1);
plot(1:size(O_ff_t_test,2),O_ff_t_test(1,:),'b-
',1:size(y,2),y(1,:),'r.');
subplot(2,3,2);
plot(1:size(O_ff_t_test,2),O_ff_t_test(2,:),'b-
',1:size(y,2),y(2,:),'r.');
subplot(2,3,3);
```

```
plot(1:size(O_ff_t_test,2),O_ff_t_test(3,:),'b-
',1:size(y,2),y(3,:),'r.');
subplot(2,3,4);
plot(1:size(O_ff_t_test,2),O_ff_t_test(4,:),'b-
',1:size(y,2),y(4,:),'r.');
subplot(2,3,5);
plot(1:size(O_ff_t_test,2),O_ff_t_test(5,:),'b-
',1:size(y,2),y(5,:),'r.');
subplot(2,3,6);
plot(1:size(O_ff_t_test,2),O_ff_t_test(6,:),'b-
',1:size(y,2),y(6,:),'r.');
```

Appendix E .- Matlab script: nn_ts_3m_2d_traf_ff_flow.m

This script is for predicting the flow of the traffic using feedforward neural network.

```
%% Clean all work environment
% traffic prediction using feedforward ANN
clear all;
close all;
clc;
%% Create ANN
% Here is how to design a neural network that predicts the target series
% from past values of inputs and targets. See narxnet for more details.
load res_vds_715898.mat;
%T=res(:,8)';
T=res_vds(:,11)';
%X=res(:,5)';
X=res_vds(:,4)';
%T={cell2mat(res(:,4))+cell2mat(res(:,5))/100}';
%[X,T] = ph_dataset;
%input number
i=1;
%number of delays for input x, and output y.
ninputs = 20;
%number of outputs of the ANN.
noutputs = 6;
while true
limInf_XT_ff = i; %infierior limit of Xtest
limSup_XT_ff = i + ninputs - 1; %superior limit of XTest
limInf_O_ff = i + ninputs; % inferior limit of Output vector (real time
measurements)
limSup_O_ff = i + noutputs + ninputs - 1; % superior limit of output
vector (real time measurements)
%if superior limit of output vector reached the end of the vector, we
have finished
%arranging the matrix for feedforwardnet
if limSup_O_ff > size(T,2)
break;
end
%arrange the vector Xtest with inputs from X, and the outputs Y from T
vector
XT_ff(i,:) = [X(1,limInf_XT_ff:limSup_XT_ff),
T(1,limInf_XT_ff:limSup_XT_ff)];
%arrange the vector Output, with values from T.
O_ff(i,:) = T(1,limInf_O_ff:limSup_O_ff);
%increase the iterator one position.
i=i+1;
end
%%
%net = feedforwardnet(13);
net = feedforwardnet(10);
X_ff_t = XT_ff'; %2X25216 X_ff_t
O_ff_t = O_ff'; %1X25216 T_ff_t
X_ff_t=cell2mat(X_ff_t); % change data type.
O_ff_t=cell2mat(O_ff_t); % change data type.
```

```
% round down to the nearest integer to split between train and test data.
nporcen=0.7;
ndat=round(size(T,2)*nporcen);
% create x vector for training the ANN
X_ff_t_train = X_ff_t(:,1:ndat);
% create y vector (output) for training the ANN
O_ff_t_train = O_ff_t(:,1:ndat);
%increase the offset by one.
ndat_1 = ndat+1;
% create x vector for testing the ANN
X_ff_t_test = X_ff_t(:,ndat_1:end);
% create y vector (output) for testing the ANN
O_ff_t_test = O_ff_t(:,ndat_1:end);
%%
% Train the feedforward ANN
tic
net = train(net,X_ff_t_train,O_ff_t_train);
toc
%view the structure of the ANN
view(net)
%%
% Obtain the output of the feedforward ANN but with testing data set.
tic
y = net(X_ff_t_test);
toc
%calculate the mean square error of the feedforward ANN.
perf = perform(net,y,O_ff_t_test)
%% Calculate RMSE
N = size(O_ff_t_test,2);
diff = O_ff_t_test - y;
diff_sqd = (diff).^2;
sum_diff_sqd = sum(diff_sqd,2);
sum_diff_sqd_N = sum_diff_sqd / N;
RMSE = sqrt(sum_diff_sqd_N);
figure(1);
plot(1:size(RMSE,1),RMSE','b*','LineWidth',2);
xlabel('Output');
ylabel('Vehicles per 5 minutes');
legend('Root mean square error');
%% Calculate MAPE
diff_abs=abs(diff);
diff_abs_div=diff_abs./O_ff_t_test;
diff_abs_div_sum = sum(diff_abs_div,2);
MAPE = diff_abs_div_sum*100/N;
figure(2);
plot(1:size(MAPE,1),MAPE','b*','LineWidth',2);
xlabel('Output');
ylabel('Percent error');
legend('Mean absolute percent error');
%% Plotresponse section
subplot(2,3,1);
plot(1:size(O_ff_t_test,2),O_ff_t_test(1,:),'b-
',1:size(y,2),y(1,:),'r.');
subplot(2,3,2);
plot(1:size(O_ff_t_test,2),O_ff_t_test(2,:),'b-
',1:size(y,2),y(2,:),'r.');
subplot(2,3,3);
```

```
plot(1:size(O_ff_t_test,2),O_ff_t_test(3,:),'b-
',1:size(y,2),y(3,:),'r.');
subplot(2,3,4);
plot(1:size(O_ff_t_test,2),O_ff_t_test(4,:),'b-
',1:size(y,2),y(4,:),'r.');
subplot(2,3,5);
plot(1:size(O_ff_t_test,2),O_ff_t_test(5,:),'b-
',1:size(y,2),y(5,:),'r.');
subplot(2,3,6);
plot(1:size(O_ff_t_test,2),O_ff_t_test(6,:),'b-
',1:size(y,2),y(6,:),'r.');
```

Appendix F .- Matlab script: sens_speed.m

This script is used to do the sensibility analysis of traffic speed feedforward ANN.

```
%% Clean all work environment
clear all;
close all;
clc;
%% Create ANN
% Here is how to design a neural network that predicts the target series
% from past values of inputs and targets. See narxnet for more details.
load('res_vds_715898.mat');
%T=res(:,8)';
T=res_vds(:,12)';
%X=res(:,5)';
X=res_vds(:,4)';
%T={cell2mat(res(:,4))+cell2mat(res(:,5))/100}';
%[X,T] = ph_dataset;
%input number
i=1;
%number of delays for input x, and output y.
ninputs = 18;
%number of outputs of the ANN.
noutputs = 6;
while true
limInf_XT_ff = i; %infierior limit of Xtest
limSup_XT_ff = i + ninputs - 1; %superior limit of XTest
limInf_O_ff = i + ninputs; % inferior limit of Output vector (real time
measurements)
limSup_O_ff = i + noutputs + ninputs - 1; % superior limit of output
vector (real time measurements)
%if superior limit of output vector reached the end of the vector, we
have finished
%arranging the matrix for feedforwardnet
if limSup_O_ff > size(T,2)
break;
end
%arrange the vector Xtest with inputs from X, and the outputs Y from T
vector
XT_ff(i,:) = [X(1,limInf_XT_ff:limSup_XT_ff),
T(1,limInf_XT_ff:limSup_XT_ff)];
%arrange the vector Output, with values from T.
O_ff(i,:) = T(1,limInf_O_ff:limSup_O_ff);
%increase the iterator one position.
i=i+1;
end
%%
%
load('speed_net_18_inputs_13_neurons_.mat');
X_ff_t = XT_ff'; %2X25216 X_ff_t
O_ff_t = O_ff'; %1X25216 T_ff_t
X_ff_t=cell2mat(X_ff_t); % change data type.
O_ff_t=cell2mat(O_ff_t); % change data type.
% round down to the nearest integer to split between train and test data.
```

```
nporcen=0.7;
ndat=round(size(T,2)*nporcen);
% create x vector for training the ANN
X_ff_t_train = X_ff_t(:,1:ndat);
% create y vector (output) for training the ANN
O_ff_t_train = O_ff_t(:,1:ndat);
%increase the offset by one.
ndat_1 = ndat+1;
% create x vector for testing the ANN
X_ff_t_test = X_ff_t(:,ndat_1:end);
% create y vector (output) for testing the ANN
O_ff_t_test = O_ff_t(:,ndat_1:end);
%%
tic
y = net(X_ff_t_test);
toc
perf = perform(net,y,O_ff_t_test)
%% numerical calculus of the derivate
%in this section the output of the ANN is obtained again,
%and a delta at the input is going to be applied in order to determine
the
%amount of change at the output to determine the mean value of this
derivative
%considering the amoutn of change in X and Y
deltas=[5 -5 10 -10 15 -15 20 -20 25 -25 30 -30];
ndeltas=size(deltas,2);
%
temp=X_ff_t_test;
net_real=net(temp); % output without modification.
ninputs_2 =size(X_ff_t_test,1);
%%
for m=1:ndeltas
for n=1:ninputs_2
temp=X_ff_t_test;
delta_in=temp(n,:)-(temp(n,:)+deltas(1,m)); % delta at input
temp(n,:)=temp(n,:)+deltas(1,m); % calculate new X + delta
net_mod=net(temp); % obtain Y of X+delta
delta_out=net_real-net_mod;% obtain amount of change in the output.
% the output Mean_dout_din, and Std_dout_din are special data
% structures three dimensional.
for k=1:noutputs
%obtain the numerical derivative aproximation
dout_din=delta_out(k,:)./delta_in;
%obtain the mean of the derivatives for the deltas applied
Mean_dout_din{k,1}(n,m)=mean(dout_din);
%obtain the standard deviation of the derivative aproximation
%for the deltas applied
Std_dout_din{k,1}(n,m)=std(dout_din);
end
end
end
%%
for k=1:noutputs
Mean_dout_din_test{k,1}=mean(Mean_dout_din{k,1}')';
Std_dout_din_test{k,1}=std(Std_dout_din{k,1}')';
end
%%
```

```
disp('Statistics for test')
for k=1:noutputs
for i=1:ninputs_2
disp(['Input ' num2str(k) ', dYdX=' num2str(Mean_dout_din_test{k,1}(i,1))
', std dYdX=' num2str(Std_dout_din_test{k,1}(i,1))])
end
end
%%
load('speed_sens_01042016.mat');
```

Appendix G .- Matlab script: sens_flow.m

This script is used to do the sensibility analysis of traffic flow feedforward ANN.

```
%% Clean all work environment
clear all;
close all;
clc;
%% Create ANN.
% Here is how to design a neural network that predicts the target series
% from past values of inputs and targets. See narxnet for more details.
load res_vds_715898.mat;
%T=res(:,8)';
T=res_vds(:,11)';
%X=res(:,5)';
X=res_vds(:,4)';
%T={cell2mat(res(:,4))+cell2mat(res(:,5))/100}';
%[X,T] = ph_dataset;
%input number
i=1;
%number of delays for input x, and output y.
ninputs = 14;
%number of outputs of the ANN.
noutputs = 6;
while true
limInf_XT_ff = i; %infierior limit of Xtest
limSup_XT_ff = i + ninputs - 1; %superior limit of XTest
limInf_O_ff = i + ninputs; % inferior limit of Output vector (real time
measurements)
limSup_O_ff = i + noutputs + ninputs - 1; % superior limit of output
vector (real time measurements)
%if superior limit of output vector reached the end of the vector, we
have finished
%arranging the matrix for feedforwardnet
if limSup_O_ff > size(T,2)
break;
end
%arrange the vector Xtest with inputs from X, and the outputs Y from T
vector
XT_ff(i,:) = [X(1,limInf_XT_ff:limSup_XT_ff),
T(1,limInf_XT_ff:limSup_XT_ff)];
%arrange the vector Output, with values from T.
O_ff(i,:) = T(1,limInf_O_ff:limSup_O_ff);
%increase the iterator one position.
i=i+1;
end
%%
%
X_ff_t = XT_ff'; %2X25216 X_ff_t
O_ff_t = O_ff'; %1X25216 T_ff_t
X_ff_t=cell2mat(X_ff_t); % change data type.
O_ff_t=cell2mat(O_ff_t); % change data type.
```

```
% round down to the nearest integer to split between train and test data.
nporcen=0.7;
ndat=round(size(T,2)*nporcen);
% create x vector for training the ANN
X_ff_t_train = X_ff_t(:,1:ndat);
% create y vector (output) for training the ANN
O_ff_t_train = O_ff_t(:,1:ndat);
%increase the offset by one.
ndat_1 = ndat+1;
% create x vector for testing the ANN
X_ff_t_test = X_ff_t(:,ndat_1:end);
% create y vector (output) for testing the ANN
O_ff_t_test = O_ff_t(:,ndat_1:end);
%%
%load data from a .mat file storing the training and simulation results.
load('flow_net_16_neurons_14_inputs.mat');
%display the structure of the ANN.
view(net);
%%
% tic and toc determines the start and the stop of the block of code
%in this case the simulation of y.
tic
y = net(X_ff_t_test);
toc
% perform obtains the mean square error (mse).
perf = perform(net,y,O_ff_t_test)
%% numerical calculus of the derivate
%in this section the output of the ANN is obtained again,
%and a delta at the input is going to be applied in order to determine
the
%amount of change at the output to determine the mean value of this
derivative
%considering the amoutn of change in X and Y
deltas=[5 -5 10 -10 15 -15 20 -20 25 -25 30 -30];
ndeltas=size(deltas,2);
%
temp=X_ff_t_test;
net_real=net(temp); % output without modification.
ninputs_2 =size(X_ff_t_test,1);
%%
for m=1:ndeltas
for n=1:ninputs_2
temp=X_ff_t_test;
delta_in=temp(n,:)-(temp(n,:)+deltas(1,m)); % delta at input
temp(n,:)=temp(n,:)+deltas(1,m); % calculate new X + delta
net_mod=net(temp); % obtain Y of X+delta
delta_out=net_real-net_mod; % obtain amount of change in the output.
% the output Mean_dout_din, and Std_dout_din are special data
% structures three dimensional.
for k=1:noutputs
%obtain the numerical derivative aproximation
dout_din=delta_out(k,:)./delta_in;
%obtain the mean of the derivatives for the deltas applied
Mean_dout_din{k,1}(n,m)=mean(dout_din);
%obtain the standard deviation of the derivative aproximation
%for the deltas applied
Std_dout_din{k,1}(n,m)=std(dout_din);
```

```
end
end
end
%%
for k=1:noutputs
Mean_dout_din_test{k,1}=mean(Mean_dout_din{k,1}')';
Std_dout_din_test{k,1}=std(Std_dout_din{k,1}')';
end
%%
disp('Statistics for test')
for k=1:noutputs
for i=1:ninputs_2
disp(['Input ' num2str(k) ', dYdX=' num2str(Mean_dout_din_test{k,1}(i,1))
', std dYdX=' num2str(Std_dout_din_test{k,1}(i,1))])
end
end
%%
load('flow_sens_02042016.mat');
```